University of Alberta

Analysis and Design of Ring-Based Transport Networks

by

George David Morley ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment
of the
requirements for the degree of *Doctor of Philosophy*

Department of *Electrical and Computer Engineering*

Edmonton, Alberta

Spring 2001

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-60329-6

Canada

University of Alberta

Library Release Form

Name of Author: *George David Morley*

Title of Thesis: *Analysis and Design of Ring-Based Transport Networks*

Degree: *Doctor of Philosophy*

Year this Degree Granted: *2001*

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

5-7 Whistler Drive, Freehold
New Jersey, USA 07728

Date: *April 6, 2001*

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled *Analysis and Design of Ring-Based Transport Networks* submitted by *George David Morley* in partial fulfillment of the requirements for the degree of *Doctor of Philosophy.*

WDGrover

Dr. Wayne D. Grover (Supervisor)
Dept. Elec. and Comp. Eng., Univ. of Alberta

Henry Owen, Georgia Inst. of Tech

Dr. Henry Owen (University External Examiner)
Elec. & Comp. Eng., Georgia Institute of Tech.

Dr. Ehab Elmallah (Dept. External Examiner)
Dept. Comp. Science, Univ. of Alberta

Dr. Bruce Cockburn (Committee Member)
Dept. Elec. and Comp. Eng., Univ. of Alberta

Ivan Fair

Dr. Ivan Fair (Committee Member)
Dept. Elec. and Comp. Eng., Univ. of Alberta

February 23, 2001

Dedication

To *Marilyn*, *Alyssa* and *Eric*.

Abstract

We introduce several new methods for the design of near-optimal telecommunication transport networks based on survivable ring architectures. Network survivability is an increasingly important issue for customers and network operators alike due to increased reliance on telecommunications services and the deployment of ultra-high capacity transmission systems. Survivable ring architectures provide a robust and cost-effective means to maintain service availability in the presence of network faults such as cable cuts, equipment failure and human error. For these reasons, they have already been widely deployed in current SONET transport networks and are a promising candidate for providing network survivability in emerging optical transport networks based on wavelength-division multiplexing.

The design of ring-based transport networks, however, is a notoriously difficult combinatorial optimization problem. For networks of any real practical size, the number of possible designs is virtually infinite and network construction costs can vary substantially from one design to another. Yet surprisingly, much this design work is still done using manual approaches that can take months to generate a single solution. Although several automated design methods have been proposed in the literature, none of these methods guarantees optimal solutions. With typical constructions costs in the range of billions of dollars, even modest improvements of only a few percent can translate into millions of dollars in savings.

We develop several improvement heuristics for a greedy heuristic that constructs a design one ring at a time. These improvement heuristics include a balanced ring loading heuristic that optimizes the routing of demands around a ring, a demand packing algorithm that routes unserved demands within the slack capacity available at each iteration, and a dithered sequencing approach that randomizes the constructive process to yield several alternative designs. We also introduce three new mathematical programming formulations of the design problem as well as a new Tabu Search meta-heuristic that explores alternative solutions by guiding a local search method.

The performance of these methods is compared using a defined set of study network models. Results show that these methods provide significant improvements in the design optimality relative to benchmark solutions. In one test case, for example, the best solution was 38% lower in cost than the previous benchmark solution. A novel lower bounding procedure and statistical inference are also used to quantify the gap from optimality of the design results.

Acknowledgements

I will be forever grateful to my wife, Marilyn, for her continual support and encouragement throughout this endeavor. I am also deeply indebted to our children, Alyssa and Eric, for their patience and understanding during the many evening and weekends that I spent writing this thesis.

I would also like to express my sincere gratitude to Dr. Wayne D. Grover for giving me the opportunity to pursue this work and for many of the basic ideas that were pursued in depth in this thesis. His numerous hours as a supervisor and advisor are greatly appreciated. The management and staff at TR*Labs* were instrumental in making the necessary funding and resources available to conduct this work and for providing a forum to test its relevancy with leading industrial sponsors such as Telus, Nortel Networks, MCIWorldcom and Sasktel. Special thanks to Linda Richens for her help in printing and distributing thesis materials. Jim Slevinsky of Telus is thanked for sharing his knowledge of the ring network design problem with me and bringing the relevant literature on the topic to my attention. John Hopkins of Nortel Networks evaluated initial versions of the software developed herein and provided valuable feedback on its features and performance. Marcia Jeremiah devoted two undergraduate co-op work terms as my assistant and contributed greatly to the implementation, testing and debugging of the software tools. Marni Mishna also made important contributions to the initial development of the software during one eight month work term with TR*Labs*. Marcia's and Marni's dedication and hard work are very much appreciated. I would also like to thank Demetrios Stamatelakis for many helpful discussions and for writing many of the Perl and Python scripts used to evaluate the results of the work. I am also grateful to the other members of my supervisory committee: Dr. B. Cockburn, and Dr. I. Fair; the external examiner, Dr. H. Owen; and the department external examiner, Dr. E. Elmallah, all of whom suggested many clarifications and corrections to the thesis.

Table of Contents

1. Introduction 1
    1.1 Thesis Organization 3
2. Definitions and Notation 6
    2.1 Introduction 6
    2.2 Mathematical Notation 6
        2.2.1 Sets and Sequences 6
    2.3 Graph Theory 7
    2.4 Combinatorial Optimization 12
    2.5 Mathematical Programming 12
    2.6 Heuristics 14
    2.7 Computational Complexity 15
        2.7.1 Classes P and NP 17
3. Transport Networks 18
    3.1 Introduction 18
    3.2 Transport Networking Technology 18
    3.3 Generic Functional Architecture 23
    3.4 Plesiochronous Digital Hierarchy 25
    3.5 Synchronous Optical Network (SONET) 27
        3.5.1 Terminal Multiplexer 30
        3.5.2 Add/Drop Multiplexer 31
        3.5.3 Digital Cross-Connect System 32
    3.6 Network Survivability 33
    3.7 Survivable Ring Architectures 36
        3.7.1 Unidirectional Path-Switched Ring 37
        3.7.2 Bidirectional Line-Switched Ring 38
    3.8 Summary 41
4. Capacity Analysis of Survivable Ring Architectures 42
    4.1 Introduction 42
    4.2 Background 42
    4.3 The Ring Sizing Problem 44
        4.3.1 Capacity Requirements for Idealized Demand Patterns 45

Table of Contents (cont'd)

4.3.2 Capacity Requirements for General Demand Patterns 47
4.4 The Ring Loading Problem 49
4.4.1 Loading Problem Formulations 51
4.4.2 Study Method 53
4.4.3 Results & Discussion 55
4.5 Summary 59
5. The Ring Network Design Problem 61
5.1 Introduction 61
5.2 Problem Description 61
5.2.1 Network Topology 62
5.2.2 Demand Matrix 62
5.2.3 Ring Technologies 62
5.2.4 Design Objective 63
5.2.5 Decision Variables 64
5.3 An Assessment of Problem Complexity 64
5.4 Other Design Considerations 66
5.4.1 Decision Environment 67
5.4.2 Demand Grooming and Hubbing 68
5.4.3 Topology Optimization 69
5.4.4 Dual Ring Interconnect 71
5.5 Summary 76
6. Related Work 77
6.1 Introduction 77
6.2 The Single Ring Network Design Problem 80
6.2.1 Gendreau et al. 80
6.2.2 Fink et al. 81
6.2.3 Lee, Ro and Tcha 81
6.2.4 Xu, Chui and Glover 82
6.2.5 Chamberland and Sansó 82
6.2.6 Chung et al. 82
6.3 The Multi-Ring Network Design Problem 83

Table of Contents (cont'd)

6.3.1 RingBuilder (Grover et al.) 83
6.3.2 Roberts 86
6.3.3 Eulerian Ring Covers (Gardner et al.) 88
6.3.4 Kennington et al. 90
6.3.5 INDT (Doshi et al.) 91
6.3.6 Net-Solver (Gardner et al.) 92
6.3.7 Bortolon et al. 94
6.3.8 Shi and Fonseka 94
6.3.9 Goldschmidt, Laugier and Olinick 97
6.3.10 Strategic Options (Wasem, Wu and Cardwell) 98
6.3.11 SONET Toolkit (Cosares et al.) 101
6.3.12 Cox et al. 102
6.4 Summary 103
7. Research Methodology 105
7.1 Introduction 105
7.2 Test Networks 105
7.3 Modeling Assumptions 110
7.3.1 Ring Technologies 110
7.3.2 Cost Model 111
7.4 Test Cases 112
7.5 Method of Analysis 113
7.5.1 Performance Metrics 113
7.6 Performance Evaluation 114
7.6.1 Empirical Testing 114
7.6.2 Lower Bounding Procedure 114
7.6.3 Statistical Inference 119
7.7 Summary 121
8. Advances on the RingBuilder Approach: RingBuilder Interactive 122
8.1 Introduction 122
8.2 Software Architecture 122
8.3 Design Synthesis Algorithm 126

Table of Contents (cont'd)

8.3.1 Basic Overview 126

8.3.2 Overview of Main Improvements 128

8.3.3 Demand Routing 128

8.3.4 Candidate Generation 129

8.3.5 Candidate Ring Evaluation 130

8.3.6 Ring Loading for Candidate Evaluation 131

8.3.7 Candidate Ring Selection 140

8.4 Improvement Heuristics 141

8.4.1 Demand Packing 142

8.4.2 Dithered Sequencing 146

8.5 Summary 148

9. Results of RingBuilder Improvements 149

9.1 Introduction 149

9.2 Comparative Study Method 149

9.2.1 Ring Loading Algorithms 149

9.2.2 Demand Packing 150

9.2.3 Dithered Sequencing 150

9.3 Results 151

9.3.1 Ring Loading Algorithms 151

9.3.2 Demand Packing 156

9.3.3 Dithered Sequencing 159

9.4 Summary 165

10. Research on Mathematical Programming applied to Multi-Ring Network Design 171

10.1 Introduction 171

10.2 Notation 171

10.3 Multi-Modular Pure Span Coverage (SCIP) 172

10.4 Fixed Charge and Routing (FCRIP) 173

10.5 Foundation Design (FDIP) 175

10.6 Comparative Study Method 176

10.7 Results 179

10.8 Summary 183

Table of Contents (cont'd)

11. A Tabu Search Meta-Heuristic for Multi-Ring Network Design 190
  11.1 Introduction 190
  11.2 Neighborhood Structure 192
  11.3 Memory Structures 192
  11.4 Search Procedure 195
  11.5 Search Diversification Strategy 197
  11.6 Comparative Study Method 198
  11.7 Results 199
  11.8 Summary 206
12. Comparative Discussion and Interpretation 209
  12.1 Introduction 209
  12.2 Solution Quality 209
  12.3 Runtimes 210
  12.4 Significance of Design Attributes 212
  12.5 Lower Bounds 213
  12.6 Statistical Inference 214
  12.7 Summary 221
13. Concluding Discussion 227
  13.1 Introduction 227
  13.2 Review of Thesis 227
  13.3 Summary of Main Contributions 229
    13.3.1 Publications 230
  13.4 Topics for Further Research 230
    13.4.1 Advanced Tabu Search Procedures 230
    13.4.2 Comparison of Demand Packing Algorithm 231
    13.4.3 Comparison of Balanced Ring Loading Algorithm 232
    13.4.4 Further Research on Topology Optimization 232
    13.4.5 Multi-Period Planning Enhancements 233
    13.4.6 Sensitivity Analysis of Design Results 234
Bibliography 235
Appendix A: Ring Loading Formulations 243

Table of Contents (cont'd)

A.1 IP3: Channel Interchange, with Demand Splitting. 243
A.2 IP4: Channel Assignment, without Demand Splitting 243
Appendix B: Optimal Ring Loading Algorithm For Hubbed Demand Patterns 245
Appendix C: Demand Patterns for Metropolitan Test Networks 248
Appendix D: Procedure for Estimating Optimal Solution Values 249
Appendix E: Cycle Finding Algorithm 252
Appendix F: Extensions to FCRIP Formulation 255
F.1 ADM Locations 255
F.2 UPSR Rings 256
Appendix G: AMPL Formulations 257
G.1 SCIP Formulation 257
G.2 FCRIP Formulation 257
G.3 FDIP Formulation 258
G.4 LBIP Formulation 259
G.5 LBRIP Formulation 260

List of Tables

Table: 3.1 North American Plesiochronous Digital Hierarchy 26
Table: 3.2 SONET Digital Signal Hierarchy 29
Table: 4.1 Gain in loading efficiency by splitting (with channel interchange) 57
Table: 6.1 Comparison of Prior Work on Multi-Ring Network Design Methods 104
Table: 7.1 Test Network Topology Statistics 107
Table: 7.2 Test Network Demand Statistics 107
Table: 7.3 Ring Technologies 111
Table: 7.4 Equipment and Facility Costs 112
Table: 7.5 Main Test Cases 112
Table: 9.1 Design Statistics for Unbalanced Ring Loading Algorithm 166
Table: 9.2 Design Statistics for Balanced Ring Loading Algorithm 167
Table: 9.3 Design Statistics for Unbalanced Ring Loading Algorithm with Demand Packing 168
Table: 9.4 Design Statistics for Balanced Ring Loading Algorithm with Demand Packing 169
Table: 9.5 Design Statistics for Balanced Ring Loading Algorithm with Dithered Sequencing 170
Table: 10.1 SCIP & FCRIP Design Parameters 177
Table: 10.2 FDIP Data Set Statistics 178
Table: 10.3 Summary of Results 179
Table: 10.4 Comparative Summary of IP Formulations 183
Table: 10.5 Computational Results for SCIP Formulation 184
Table: 10.6 Computational Results for FCRIP Formulation 185
Table: 10.7 Computational Results for FDIP Formulation 186
Table: 10.8 Design Statistics for SCIP Formulation 187
Table: 10.9 Design Statistics for FCRIP Formulation 188
Table: 10.10Design Statistics for FDIP Formulation 189
Table: 11.1 Test Parameters Settings 198
Table: 11.2 Total Design Cost for Several Add and Drop Tenures 204
Table: 11.3 Detailed Computational Results for Tabu Search Meta-Heuristic 207
Table: 11.4 Design Statistics for Tabu Search Meta-Heuristic 208
Table: 12.1 Point and Interval Estimates of Global Optimal Solutions 215
Table: 12.2 Summary of Total Design Cost Results 222

List of Tables (cont'd)

Table: 12.3 Summary of Runtime Results (sec.) 223
Table: 12.4 Correlation between Design Attributes and Total Design Cost 224
Table: 12.5 Lower Bounds based on Shortest Path Routing 225
Table: 12.6 Lower Bounds with Route Optimization 226
Table: D.1 Solution Data for Test Case 12....249

List of Figures

Figure 2.1. Set notation. 7

Figure 2.2. Diagrams of undirected and directed graphs 8

Figure 2.3. Examples of (a) walk, (b) trail and (c) path. 9

Figure 2.4. Examples of Eulerian and Hamiltonian graphs. 9

Figure 2.5. Examples of a (a) two-vertex connected graph and (b) two-edge connected graph. 10

Figure 2.6. Examples of a tree, forest, spanning tree and spanning forest. 10

Figure 2.7. An example of a fundamental set of cycles 11

Figure 3.1. Layering of service and transport networks. 19

Figure 3.2. Classification of transport networks. 20

Figure 3.3. Layered view of a transport network (client/server association) 24

Figure 3.4. Partitioned view of a transport network 25

Figure 3.5. SONET STS-1 frame 28

Figure 3.6. SONET section, line and path equipment. 30

Figure 3.7. Functional block diagram of a TM. 30

Figure 3.8. Functional block diagram of an ADM. 31

Figure 3.7. Functional block diagram of a B-DCS. 32

Figure 3.8. Classification of survivable network architectures. 34

Figure 3.9. Two-fibre UPSR protection switching operation 37

Figure 3.10. Four-fibre BLSR protection switching operation. 38

Figure 3.11. Two-fibre BLSR protection switching operation. 39

Figure 3.12. An example of BLSR ring loading. 41

Figure 4.1. An example of channel assignment in a BLSR 44

Figure 4.2. Idealized demand patterns 45

Figure 4.3. Relative BLSR demand carrying capacity for idealized demand patterns. 47

Figure 4.4. Gain in loading efficiency due to channel interchange. 56

Figure 4.5. Scatter plot of loading efficiency vs. demand pool size for the 5 node/48 channel configuration, with a mesh demand pattern and channel interchange. 58

Figure 4.6. Scatter plot of loading efficiency vs. demand pool size for the 10 node/48 channel configuration, with a mesh demand pattern and channel interchange. 58

Figure 5.1 Functional diagram of multi-ring network design problem 61

List of Figures (cont'd)

Figure 5.2. Upper bound on the number of possible designs. 66
Figure 5.3. An example of demand routing in multi-ring networks. 70
Figure 5.4. Matched node drop & continue inter-ring transfer arrangement for BLSRs. 72
Figure 5.5. Matched node drop & continue inter-ring transfer arrangement for UPSRs. 73
Figure 5.6. Dual feeding inter-ring transfer arrangement. 74
Figure 5.7. An example ring connectivity graph. 75
Figure 6.1. Classification of Ring Network Design Problems. 77
Figure 6.2. Effect of ring modularity on transmission capacity requirments. 78
Figure 6.3. Effect of topological layout on transmission capacity requirments. 79
Figure 6.4. Simulated Annealing generate operations. 87
Figure 6.5. Decomposing an Eulerian graph into cycles. 88
Figure 6.6. Converting a non-Eulerian graph to an Eulerian graph. 89
Figure 6.7. Net-Solver iteration strategies 93
Figure 6.8. An example of a hierarchical ring network. 95
Figure 6.9. Strategic Options: An example of a multi-period demand bundling choices 99
Figure 6.10. Strategic Options: An example of ring selection 100
Figure 6.11. Strategic Options: An example of the design alternatives for a potential ring 100
Figure 6.12 Strategic Options: An example of a multi-period multiplex cost algorithm. 101
Figure 7.1. Topology of the metropolitan test networks. 106
Figure 7.2. Topology of the long-haul test networks. 106
Figure 7.3. Demand distribution for Net15 108
Figure 7.4. Demand distribution for Net20 108
Figure 7.5. Demand distribution for Net32 109
Figure 7.6. Demand distribution for Net43 109
Figure 7.7. Ring parity condition: (a) working span loads, (b) possible ring cover. 115
Figure 7.8. Ring balance condition. (a) working span loads, (b) possible ring cover. 117
Figure 7.9. Distribution of suboptimal solutions relative to the optimal solution 120
Figure 8.1. RingBuilder Interactive software architecture. 123
Figure 8.2. RingBuilder Interactive screen capture. 123
Figure 8.3. Diagram of main classes in RingBuilder Interactive 124
Figure 8.4. Flow chart of the basic RingBuilder algorithm. 127

List of Figures (cont'd)

Figure 8.5. Main steps in unbalanced ring loading procedure. 133

Figure 8.6. Example of the unbalanced ring loading procedure. 134

Figure 8.7. Example of the balanced ring loading procedure. 139

Figure 8.8. Effect of exponent *x* on the selection probability. 141

Figure 8.9. Simplified the demand packing algorithm. 143

Figure 8.10. An example of a ring connectivity graph. 144

Figure 8.11. Illustration of the basic Dithered Sequencing meta-heuristic. 147

Figure 9.1. Results for unbalanced and balanced ring loading algorithms. 152

Figure 9.2. Network design for test case 2 (Net15, 4B48) using the unbalanced ring loading algorithm. 153

Figure 9.3. Network design for test case 2 (Net15, 4B48) using the balanced ring loading algorithm. 153

Figure 9.4. Effect of the ADM discount factor on total design cost for test case 5 (Net20, 4B48) and test case 6 (Net20, 4B48 & 4B192) using the balanced ring loading algorithm. 155

Figure 9.5. Effect of the adjusted transport efficiency exponent on total design cost for test case 5 (Net20, 4B48) and test case 6 (Net20, 4B12 & 4B48) for Net20 using the balanced ring loading algorithm. 156

Figure 9.6. Results for unbalanced and balanced ring loading algorithms with demand packing. 157

Figure 9.7. Total design cost versus demand packing ADM discount factor for test case 5 (Net20, 4B48) and 6 (Net20, 4B12 & 4B48). 158

Figure 9.8. Total design cost versus greedy selection threshold for test case 5. 159

Figure 9.9. Design cost versus probabilistic selection exponent for test case 5 with a greedy selection threshold. 160

Figure 9.10. Plot of the (a) absolute and (b) relative minimum, maximum and mean transport efficiencies of the elite candidate rings as a function of design iteration. 161

Figure 9.11. Probability distribution of the the top ten candidate rings as a function of the probabilistic selection exponent x for the (a) first and (b) final iterations. 161

List of Figures (cont'd)

Figure 9.12. Relative frequency histogram of the total design cost (relative to the baseline greedy soluton) for test cases 1-9 using the probabilistic selection method and the balanced ring loading algorithm. 162

Figure 9.13. Design sequences for test case 5 using parameter settings (3,3,3). 163

Figure 9.14. Effect of different Dithered sequencing parameter settings on total design cost. 163

Figure 9.15. Effect of no. of branches per elite sequence on total design cost. 164

Figure 9.16. Total design cost versus greedy selection threshold for test case 5 (Net20, 4B48) 164

Figure 10.1. Effect of the number of paths on FCRIP total design cost for test case 1 (Net15, 4B12). 182

Figure 11.1. A graphical illustration of the Tabu Search meta-heuristic. 190

Figure 11.2. Basic version of Tabu Search meta-heuristic 191

Figure 11.3. Flow chart of basic Tabu Search meta-heuristic. 195

Figure 11.4 Results for Tabu Search meta-heuristic. 199

Figure 11.5 Example of TS search trajectory. 200

Figure 11.6 Initial starting solution for Net20. 201

Figure 11.7 Tabu Search solution for Net20. 201

Figure 11.8 Span utilization (working load/capacity) in DS3s for Net20. 202

Figure 11.9 Histogram of the slack capacity for Net20. 203

Figure 11.10. Plot of total design cost vs. (a) drop depth and (b) restart window. 205

Figure 11.11. Plot of total design cost vs. (a) restart penalty and (b) ADM discount factor. 205

Figure 12.1. Relative Performance of design methods (% gap) relative to the best solution for all test cases 209

Figure 12.2. Plot of runtime vs. problem size for baseline RingBuilder heuristic and Tabu Search meta-heuristic. 211

Figure 12.3 Plot of runtime vs. problem size for SCIP formulation. 211

Figure 12.4 Star plot of the correlation squared between several design attributes and total design cost for test cases 5 and 7. 212

Figure 12.5. Histogram of total design cost for test case 1 (Net15, 4B12). 215

Figure 12.6 Histogram of total design cost for test case 2 (Net15, 4B48). 216

Figure 12.7 Histogram of total design cost for test case 3 (Net15, 4B12 + 4B48). 216

List of Figures (cont'd)

Figure 12.8. Histogram of total design cost for test case 4 (Net20, 4B12)........ 217

Figure 12.9. Histogram of total design cost for test case 5 (Net20, 4B48)........ 217

Figure 12.10. Histogram of total design cost for test case 6 (Net20, 4B12 + 4B48)........ 218

Figure 12.11. Histogram of total design cost for test case 7 (Net32, 4B48)........ 218

Figure 12.12. Histogram of total design cost for test case 8 (Net32, 4B192)........ 219

Figure 12.13. Histogram of total design cost for test case 9 (Net32, 4B48 + 4B192)........ 219

Figure 12.14. Histogram of total design cost for test case 10 (Net43, 4B48)........ 220

Figure 12.15. Histogram of total design cost for test case 11 (Net43, 4B192)........ 220

Figure 12.16. Histogram of total design cost for test case 12 (Net43, 4B48 + 4B192)........ 221

Figure B.1. Ring loading algorithm for pure hubbed demand pattern........ 246

Figure D.1 Probability plot for test case 12........ 250

Figure D.2 Correlation squared vs. location parameter........ 251

Figure E.1. The recursive depth-first search procedure for finding cycles........ 253

Figure E.2 The main procedure for finding all cycles in a graph........ 254

1. Introduction

Telecommunications services form an integral part of our information-centric society and play a significant, if not vital, role in our local, national and global economies. At the core of today's telecommunications infrastructure are high-capacity transport networks that convey virtually all telecommunications services (e.g., voice, video, data and Internet services) at some point or another along their journey from origin to destination. These "backbone" networks are usually composed of fibre optic transmission systems that concentrate traffic onto a surprisingly small number of high-capacity routes. For example, state-of-the-art Synchronous Optical Network (SONET) transmission systems operating at 10 Gbps can carry the equivalent of more than 120,000 voice conversations on a single pair of fibres. When combined with recent wavelength-division multiplexing (WDM) techniques, the aggregate capacity of a single fibre pair can be as high as 400 Gbps — the equivalent of more than 5 million simultaneous voice calls. With some cables containing upwards of 48 fibre pairs, a single cable cut can have disastrous and far-reaching effects. Although cable cuts are by far the most prevalent form of failure, transport networks are also susceptible to several other types of failures including human error, software bugs, fires, earthquakes and flooding. The FCC has reported that in the United States network outages affecting more than 30,000 customers occur every one to two days and take five to ten hours on average to repair [JHV99]. The social and economic impact of these outages can be substantial.

For these reasons, network *survivability* has become an important issue for customers and network operators alike. Many customers are now demanding 100% service availability from their network operator. In response, special precautions (e.g., duplicate equipment, reinforced cables, fire suppression equipment, etc.) are usually taken to help avoid network failures. Protection and restoration mechanisms have also been developed to reroute traffic around network failures using spare transmission resources. Two basic approaches for protecting transport networks are under study today. These are mesh restoration and survivable ring architectures [Gro97c]. In mesh restoration, the traffic affected by a failure is restored by establishing alternate paths through a layer of spare capacity that is shared by all services. In contrast, survivable rings rely on dedicated transmission facilities to protect only those sites served by the ring. While mesh restoration achieves the lowest redundancy in transmission capacity needed for 100% restorability, ring architectures are often preferred in practice because of their simpler and faster switching mechanism (50 ~150 milliseconds). Also, despite their greater capacity requirements, rings can be more economical than mesh networks, particularly in metropolitan area networks where nodal costs usually dominate over distance-dependent costs for fibre and regenerators. For these reasons, SONET-based rings have already been widely deployed and the same logical architectures are promising and obvious candidates for the emerging all-optical network.

Although the basic protection mechanism is relatively simple, the design of transport networks using survivable rings is an extremely difficult optimization problem. Solving this problem is especially difficult because it involves simultaneously finding an optimal routing pattern and a set of rings that satisfies all network demands at minimum cost. Furthermore, for each ring in the solution the following decision variables must be specified: the nodes to include in the ring, the fibre routing between the nodes, the ring type and capacity, the assignment of demands, and the nodes at which demands transit from one ring to another enroute from origin to destination. For networks of any real practical size, the number of possible designs is virtually infinite and network construction costs can vary substantially from one design to another. Yet surprisingly, much this design work is still done using manual approaches that can take months to generate a single solution. Although several automated design methods have been proposed in the literature, none of these methods guarantees optimal solutions. With typical constructions costs typically in the range of billions of dollars, even modest improvements of only a few percent can translate into millions in savings.

This provides the basic motivation for the research presented in this thesis. The primary goal of this work is to better understand the factors that influence the design of ring-based transport networks and to develop improved methods and algorithms for automating the design process. As part of this work, we investigate several policy and technology alternatives related to the demand loading on bidirectional line-switched rings. We also develop several improvement heuristics for a greedy heuristic that constructs a network design one ring at a time. These improvement heuristics include a balanced ring loading heuristic that optimizes the routing of demands around a ring, a demand packing algorithm that routes unserved demands within the slack capacity available at each iteration, and a dithered sequencing approach that randomizes the constructive process to yield several alternative designs. We also introduce three new mathematical programming formulations of the design problem and a new Tabu Search meta-heuristic that explores alternative solutions by guiding a local search method.

The performance of these methods is compared using a defined set of study network models. Results show that these methods provide significant improvements in the design optimality relative to benchmark solutions. In one test case, for example, the best solution was 38% lower in cost than the previous benchmark solution. A novel lower bounding procedure and statistical inference are also used to quantify the optimality of the design results.

1.1 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 begins by introducing basic notation, concepts and terminology from several areas of study that are used throughout this thesis. This includes relevant concepts and results from graph theory, combinatorial optimization, mathematical programming and complexity theory.

Chapter 3 contains additional background information on transport network technology, equipment and networks. The chapter begins by discussing the different classes of transport networks with particular emphasis on the functional characteristics of circuit-switched transport networks, which are the main focus of this work. Two current transport networking technologies, the Plesiochronous Digital Hierarchy (PDH) and Synchronous Optical Networks (SONET) are then described in detail. This is followed by an overview of the three main protection and restoration methods and a detailed description of the two most common types of survivable ring architectures: the unidirectional path-switched ring (UPSR) and the bidirectional line-switched ring (BLSR).

In Chapter 4, we consider capacity-related aspects of survivable ring architectures and introduce two important subproblems in the design of multi-ring networks: the Ring Sizing Problem, which involves finding the minimum size (capacity) ring for a specified set of demands, and the Ring Loading Problem, which involves finding the best subset of demands to load onto a ring of fixed size (capacity). Analytical models for the Ring Sizing Problem are presented for a variety of idealized demand patterns. This is followed by a survey of prior work on the much-studied generalized Ring Sizing Problem. We then introduce the Ring Loading Problem and develop mathematical programming formulations for several variations of the problem. Simulations using realistic demand patterns are then used to quantify the effects of two policy and technology choices (i.e., demand splitting and time-slot interchange) on ring loading efficiency. The findings of this study have important implications on the work in subsequent chapters.

Chapter 5 provides a formal definition of the multi-ring design problem and discusses several aspects of the problem inputs and basic design assumptions. This is followed by an analysis of the computational complexity of the problem and a derivation of an upper bound on the number of possible designs. Several other related design considerations, such as topology optimization and dual ring interconnection issues, are also discussed at length.

Chapter 6 surveys the prior work on ring-related network design problems. We begin by identifying the main classes of ring design problems and the basic solution techniques proposed to date. This survey covers work on both the single and multi-ring network design problems. In total, some eighteen different methods are described. This chapter concludes with a summary of the differences between these design methods and the work conducted in this thesis.

In Chapter 7, we describe the study method used to evaluate the performance of the design methods developed in this thesis. First, the characteristics of the test networks and their demand patterns are described in detail. This is followed by a description of the basic modeling assumptions, ring types, cost model and test scenarios that were used to obtain the results. The metrics used to quantify the performance of the design methods are then presented along with a description of the techniques used to assess both relative and absolute solution quality.

Chapters 8 through 11 describe the three main design approaches that were proposed, implemented and tested in this thesis. Chapter 8 provides a detailed description of a sophisticated network planning tool, called RingBuilder Interactive, that was developed to support this work. The chapter begins with a brief overview of the software architecture and the main features and capabilities of RingBuilder Interactive. This is followed by a description of the basic algorithmic framework used to synthesize network designs. A detailed description is then provided for each of the main algorithms that comprise the design method as well as two proposed improvement heuristics. In Chapter 9, the main algorithms and improvement heuristics are tested over a range of parameter values and the study method and results are presented.

In Chapter 10, we develop three mathematical programming formulations for the multi-ring network design problem. Each of these formulations represents a different tradeoff along the continuum between model detail and tractability. While these formulations have certain claims to optimality within their logical problem models, computational constraints usually prevent finding strictly optimal solutions. Nonetheless, they do provide relatively good solutions that serve as useful benchmarks for the other design methods.

In Chapter 11, we develop a novel Tabu Search procedure for the multi-ring network design problem. Tabu Search is a meta-heuristic that guides a local search procedure to explore regions in the solution space beyond a local optimum. The chapter begins with an overview of the basic procedure, followed by a description of the problem specific memory structures and operations developed here. The procedure is tested over a range of parameter settings and the results are presented at the end of the chapter.

Chapter 12 provides a comparative discussion of the three design methods presented in Chapters 8 through 11 and an interpretation of the main findings. The design methods are compared in terms of solution quality and runtime. A lower bounding procedure and statistical inference are also used to quantify the absolute performance of the design methods.

Chapter 13 concludes with a review of the thesis, a summary of the research results and some recommendations for further work on this topic.

2 Definitions and Notation

2.1 Introduction

This chapter introduces basic notation, concepts and terminology that are relevant to the work presented in this thesis. This material is provided primarily for review and reference. We start by defining the mathematical notation used in this thesis. This is followed by an overview of some basic concepts and results from several related areas of study. These include graph theory, combinatorial optimization, mathematical programming, algorithm analysis and complexity theory.

2.2 Mathematical Notation

This section introduces some basic mathematical definitions and symbols used in this thesis. In general, we try to use each symbol to denote only one quantity but, because there are only 26 Latin letters, this is not always possible. We will usually define symbols where they are first used. The two most common mathematical concepts that are used are sets and sequences. Definitions and notations for sets and sequences are provided in the following subsections.

2.2.1 Sets and Sequences

The concept of a set arises frequently in formulating and solving network planning problems. A *set* is an unordered ensemble or collection of objects [AKL84]. The objects that form a set are called *elements*. A set is completely specified by the elements that belong to it, without regard to the relations between the elements. For example, a set containing five objects arranged in circle is the same as a set containing the same five objects arranged in a line. Thus, there is no concept of order in a set. The elements of a set are drawn from a population known as a *base type*. Each element of a set is either a primitive element of the base type or is a set. There is also no concept of duplication in a set; all elements in the same set are distinct from one another and an element either belongs to the set or not. The symbols used here to express sets and their relationships are shown in Figure 2.1.

To illustrate the use of this notation, consider two sets $M = \{a, b, c\}$ and $N = \{b, e, f, g\}$. Because M has three elements, $|M| = 3$. Likewise, $|N| = 4$ because N has four elements. The union of M and N is $M \cup N = \{a, b, c, e, f, g\}$. The intersection of M and N is the set of elements that appear in both M and N, which is $M \cap N = \{b\}$. The set difference of M and N is $M - N = \{a, c\}$. The symmetric difference of M and N is the set of elements appearing in M and N but not in both, which is $M \oplus N = \{a, c, e, f, g\}$.

A concept that is closely related to sets is that of a sequence. A *finite sequence* of length n is an ordered set containing n objects. A finite sequence will also be denoted using braces, e.g.

$\{x_1, x_2, \ldots, x_n\}$. Unlike a set, a sequence may contain duplicates that are distinct elements, e.g. $x_2 = x_n$. Both sets and sequences are used extensively in the following subsections.

$M = \{a, b, c\}$	set M composed of elements a, b and c.
$a \in M$	a is an element of set M.
$n \notin M$	n is not an element of set M.
$\forall m \in M$	for each element m in set M.
$\emptyset$	the null or empty set.
$\|M\|$	the *cardinality* or number of elements in set M.
$\{x \mid x \text{ is a positive integer}\}$	*set former*: e.g. all x such that x is a positive integer.
$A \subseteq B$, $B \supseteq A$	set A is a subset of set B, set B is a superset of set A.
$A \cup B$	*union*: all elements in either A or B.
$A \cap B$	*intersection*: all elements in both A and B.
$A - B$	*difference*: all elements in A that are not in set B.
$A \oplus B$	*symmetric difference*: all elements in either A or B, excluding those elements in both A and B, i.e. $\{x \mid x \in A \cup B, x \notin A \cap B\}$.

Figure 2.1. Set notation (adapted from [Sha98]).

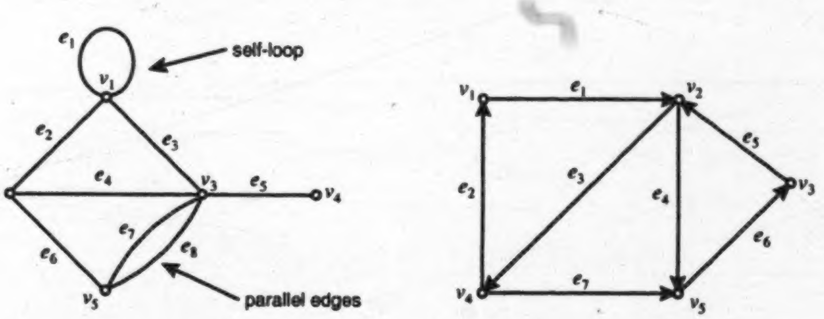
2.3 Graph Theory

It is often convenient to represent transport networks graphically by a set of points and a set of lines connecting pairs of points. Typically, the points represent network *nodes* (e.g., switching locations or equipment enclosures) and the lines between them represent network *spans* (e.g., transmission facilities or cable ducts). An abstraction of this concept is a graph.

A *graph* $G = (V, E)$ consists of a finite set of *vertices* $V = \{v_1, v_2, \ldots\}$ and a set of *edges* $E = \{e_1, e_2, \ldots\}$ such that each edge in E joins a pair of vertices in V [BoM76]. Two vertices, $u \in V$ and $v \in V$, are *adjacent* if they are joined by an edge $e = \{u, v\} \in E$. Such an edge is said to be *incident* on the vertices *u* and *v*, which are also called the *end* vertices of *e*. Two edges are said to be *adjacent* if they are incident on a common vertex and *parallel* if they are incident on the same pair of end vertices. An edge that begins and ends on the same vertex is called a *self-loop*. A graph is *simple* if it has no parallel edges or self-loops. Whereas, a graph with parallel edges is called a *multigraph*.

In some graphs, a number w_{uv} may be associated with every edge $\{u, v\}$. Such a graph is called a *weighted graph* and the number w_{uv} is called the *weight* of edge $\{u, v\}$. In transport networks these weights are typically used to represent such things as cost, distance, utilization or capacity. When these weights are used to represents capacities, the graph is usually called a *capacitated graph*. Capacitated graphs are often used for solving problems that involve the flow of commodities through a network.

An edge is *directed* if its vertices $\{u, v\}$ are an ordered pair. A directed edge is drawn by a line segment with an arrowhead indicating the direction. A graph with directed edges is called a *directed graph* or *digraph*. A graph is *undirected* if all its edges are not directed. Unless otherwise stated, we represent transport networks as undirected graphs or, equivalently, as networks in which transmission capacity is assumed to be bidirectional and symmetric. A diagram of an undirected graph and a directed graph are shown Fig. 2.2(a) and 2.2(b), respectively.



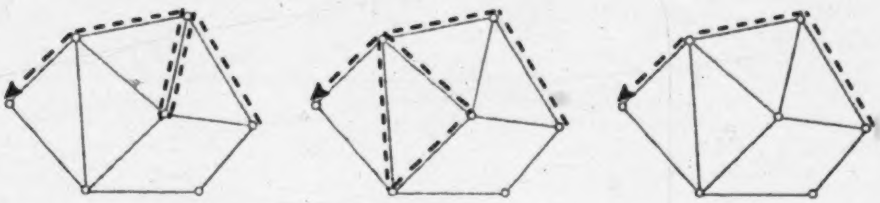
(a) An undirected graph.

(b) A directed graph.

Figure 2.2. Diagrams of undirected and directed graphs.

The number of vertices and edges in graph $G = (V, E)$ is denoted by $|V|$ and $|E|$, respectively. The number of edges incident on a vertex is called the *degree* of the vertex. For example, in Fig. 2.2(a) vertex v_2 has a degree of three and vertex v_3 has a degree of five.

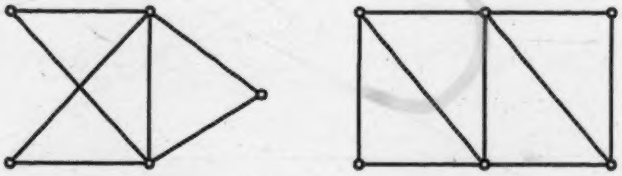
A *walk* in G is a sequence of adjacent edges $W = \{(v_1,v_2), (v_2,v_3), \ldots, (v_{k-2},v_{k-1}), (v_{k-1},v_k)\}$. The vertices v_1 and v_k are called the *origin* and *destination* of the walk, respectively. The number of edges in the walk, $k-1$, is called the *length* of the walk. A *trail* is a walk in which the edges are distinct. If the vertices in a trail are also distinct, it is called a *path*. Note that the term "path" is also used to denote connections in transport networks. In most cases, however, the intended meaning should be clear from the context. Examples of a walk, trail and path are shown in Fig. 2.3.



(a) Walk. (a) Trail. (a) Path.

Figure 2.3. Examples of (a) walk, (b) trail and (c) path.

A walk is *closed* if its origin and destination are the same. A closed trail is called a *tour*. A tour that traverses all edges in a graph is known as an *Eulerian tour*. A graph containing an Eulerian tour is called an *Eulerian graph*. It can be shown that a connected undirected graph has an Eulerian tour if and only if it has no vertices of odd degree [AKL84]. If the vertices in a tour are distinct it is called a *cycle*. A cycle that connects all of the vertices in a graph is called a *Hamiltonian* cycle. If a graph contains an Hamiltonian cycle it is called an *Hamiltonian graph*. Unlike an Eulerian graph, there is no easy test to determine whether a graph contains a Hamiltonian cycle. Examples of Eulerian and Hamiltonian graphs are shown in Fig. 2.4.

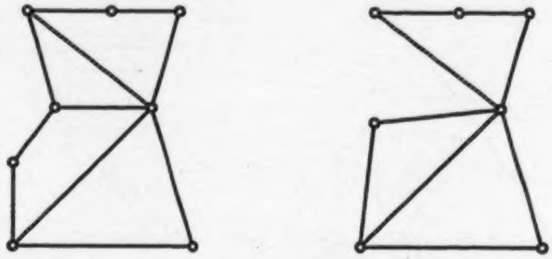


(a) Eulerian Graph (b) Hamiltonian graph

Figure 2.4. Examples of Eulerian and Hamiltonian graphs.

Note that the Eulerian graph in Fig. 2.4(a) does not contain a Hamiltonian cycle. Similarly, the Hamiltonian cycle in Fig. 2.4(b) does not contain an Eulerian tour because it has two vertices of odd degree. In general, however, a graph may be both Eulerian and Hamiltonian because these properties are not mutually exclusive.

A graph $G' = (V', E')$ is a subgraph of G if $V' \subseteq V$ and $E' \subseteq E$. Two vertices in G are said to be *connected* if there exists a path between them in G. An undirected graph G is *connected* if there exists at least one path between every pair of vertices in G. A subgraph $G' = (V', E')$ of G is called a *connected component* of G if it is connected and there is no other connected subgraph of G that contains G'. A graph is *two-edge connected* if it has at least two edge-disjoint paths between every pair of vertices. Two paths are edge disjoint if no edge is a member of both paths. A graph is *two-vertex connected* or *biconnected* if it has at least two vertex-disjoint paths between every pair of nodes. Figure 2.5 shows an example of a two-vertex connected and two-edge connected graph.

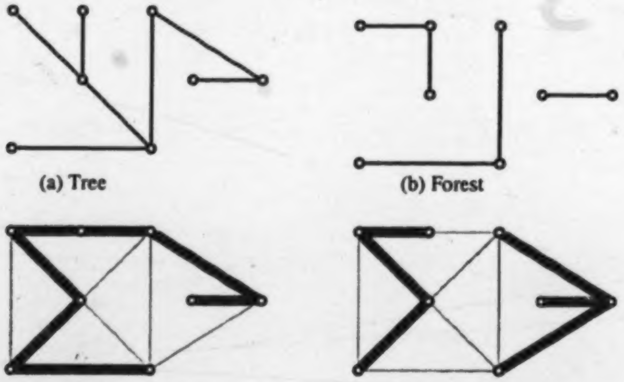


(a) Two-vertex-connected Graph (b) Two-edge-connected Graph

Figure 2.5. Examples of a (a) two-vertex connected graph and (b) two-edge connected graph.

Connectivity is an important aspect in the study of survivable transport networks. Two-edge connectivity, for example, is a necessary condition for a network to be able to survive any single edge (span) failure. Likewise, two-vertex connectivity is required to able to survive any single span and vertex (node) failure.

An *acyclic* graph is one that contains no cycles. A *tree* is a connected acyclic graph. A *forest* is a set of trees. Examples of a tree and forest are shown in Fig. 2.6(a) and Fig. 2.6(b), respectively.



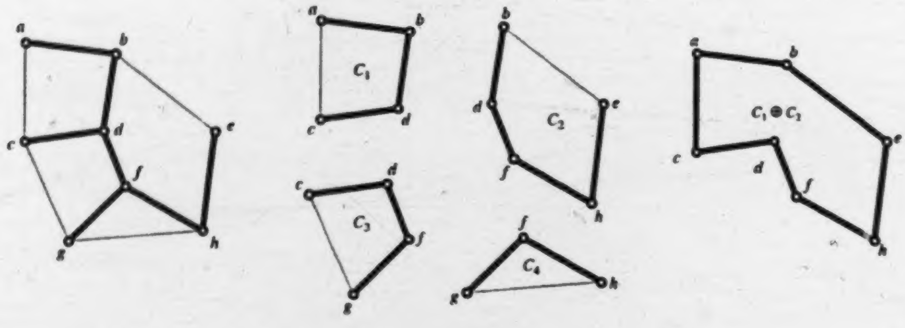
(a) Tree (b) Forest

(c) Spanning tree in a graph (d) Spanning forest in a graph

Figure 2.6. Examples of a tree, forest, spanning tree and spanning forest.

In a connected, undirected graph there is at least one path between every pair of vertices. The absence of a cycle in a connected, undirected graph implies that there is exactly one path between every pair of vertices. A *spanning tree* is a subgraph of G that contains every vertex in G. If G is not connected, the set of trees for each connected component is called a *spanning forest*. Examples of a spanning tree and spanning forest are shown in Fig. 2.6(c) and Fig. 2.6(d), respectively.

An important problem in the design of ring-based transport networks is finding the set of cycles within a network graph. This is because each cycle represents a node-disjoint (and span-disjoint) fibre route over which a ring can be created. Note that if the fibre route is not node-disjoint, then there is at least one single point of failure in the ring. An interesting property of an undirected graph is that the entire set of all cycles can be constructed from a *fundamental set of cycles* of the graph. That is, a fundamental set of cycles completely determines the cycle structure of a graph because every cycle can be created from a combination of fundamental cycles. A set of fundamental cycles can be found from a spanning tree $\{V, T)$ of a connected undirected graph $G = (V, E)$. Any edge in E but not in T will create exactly one cycle when added to T. Because every spanning tree of G has $|V| - 1$ edges, the number of fundamental cycles is $|E| - |V| + 1$ [MaD76]. Let $F = \{C_1, C_2, \ldots, C_{|E| - |V| + 1}\}$ be a fundamental cycle set of G. Then any cycle in G can be written as $((C_{i_1} \oplus C_{i_2}) \oplus \ldots) \oplus C_{i_t}$, where $\oplus$ is the symmetric difference operation. For example, Fig 2.7 shows a graph and the set of fundamental cycles obtained from the spanning tree of the graph (in boldface).



(a) (b) (c)

Figure 2.7. An example of a fundamental set of cycles. (a) A graph G and spanning tree T (in boldface), (b) the fundamental cycle set from T, (c) generating another cycle from two fundamental cycles (adapted from [RND77]).

The cycle in Fig. 2.7(c) is formed by the symmetric difference of cycles C_1 and C_2, i.e.

$C_1 \oplus C_2$. Note that not every combination of fundamental cycles forms a cycle. For example, the symmetric difference of cycles C_1 and C_4 comprises two disjoint cycles. The fact that all cycles can be represented from any basis set of fundamental cycles is exploited by some ring design methods.

2.4 Combinatorial Optimization

The majority of problems considered in this thesis belong to a class of problems known as *combinatorial optimization problems*. A combinatorial optimization problem is a decision problem that involves selecting discrete alternatives, i.e. where the solution is a set or a sequence of integers or other discrete objects [Kno89]. Two properties of combinatorial optimization problems are:

(i) the number of feasible solutions usually increases rapidly as the size of the input increases.

(ii) it is easy to construct a feasible solution.

A classic example of a combinatorial optimization problem is the *Travelling Salesman Problem* (TSP). This problem involves finding a tour that visits n cities where the travel time between each of the pairs of cities is known and the objective is to minimize the total travel time. Like many combinatorial optimization problems, TSP is easy to state but difficult to solve because there are so many alternatives to consider. A naive approach to solving an instance of this problem is to list all possible solutions, evaluate their objective functions and pick the best solution. If n is large, however, evaluating all possible solutions quickly becomes impractical because the number of possible tours is $(n-1)!$. For example, a TSP with only 20 cities has more than 10^{17} possible tours. Even if one could evaluate a million tours per second it would take over 3.8 millennia to enumerate all possible tours. And this is a relatively small TSP problem!

Another classic combinatorial optimization problem is the *Knapsack Problem*. Here a set of items is available to be packed into a knapsack with capacity c units. Each item i has a utility u_i and requires w_i units of capacity. The problem is to determine the subset of items I that maximizes the total utility without exceeding the capacity of the knapsack. Both the TSP and the Knapsack problem are closely related to several subproblems in ring network design that are discussed in subsequent chapters.

2.5 Mathematical Programming

Combinatorial optimization problems belong to broader class of problems known as *mathematical programming problems*. The term "programming" is used here to describe the planning or scheduling of activities in a large organization and is not to be confused with "computer programming." Mathematical programming problems consist of three main components: a set of unknowns or decision variables, an objective function and a set of constraints. A *decision variable* is the amount or level of each activity to be determined. The *objective function* is a function of the decision variables (e.g., cost, profit) that we want to either minimize or maximize. And the constraints are a set of equations that restrict the decision variables to values that would be considered an acceptable or *feasible* solution to the problem. Mathematical programming problems can be expressed in algebraic form as follows:

$$\textit{Minimize:} \quad f(x) \tag{2.1}$$

$$\textit{Subject to:} \quad g_i(x) \geq b_i, \qquad i = 1, \ldots, m \tag{2.2}$$

$$h_j(x) = c_j, \qquad j = 1, \ldots, n \tag{2.3}$$

where x is a vector of decision variables, $f(x)$ is the objective function and $g_i(x)$ and $h_j(x)$ together with b_i and c_j, respectively, form the *system of constraints*. Here the *sense of optimization* is minimization but it could just as easily have been maximization with obvious changes to the objective function and constraints. Similarly, the inequality in any of the constraints in Eq. (2.2) can be reversed simply by multiplying both sides of the equation by minus one.

If the objective function and constraints are restricted to linear functions and the decision variables are allowed to take fractional (continuous) values, the problem is called a *linear program* (LP) and the process of solving it is called *linear programming*. In this case, the coefficients on the left-hand side of the system of constraints is called the *coefficient matrix*. Many combinatorial optimization problems can be formulated as LPs in which some or all of the decision variables are restricted to whole number, or integral, values. An LP with all integer decision variables is called an *integer program* (IP). An LP problem with a combination of linear and continuous decision variables is called a *mixed integer program* (MIP). The Knapsack Problem can be expressed by the following IP:

$$\textit{Maximize:} \quad \sum_{i \in I} u_i \cdot X_i \tag{2.4}$$

$$\textit{Subject to:} \quad \sum_{i \in I} w_i \cdot X_i \leq c \tag{2.5}$$

$$X_i \in \{0, 1\}, \qquad \forall i \in I \tag{2.6}$$

where X_i is a binary decision variable that equals one if item i is packed into the knapsack and zero otherwise. In the above formulation, we use lower case letters with and without a subscript for model parameters such as utility, weights and capacity and an upper case letter with subscript for the decision variables. An upper case letter is also used to denote the set of items but it is easy to distinguish between the two from the context. This convention for labelling decision variables, sets and parameters is used throughout the remainder of this thesis.

The formulation of the Knapsack Problem is a fairly straight-forward. Formulating some combinatorial optimization problems as IP models, however, requires a fair bit of ingenuity. One disadvantage of IP models is that the time required to find the optimal solution may become excessive as the problem size increases. In some special cases, however, the optimal solution to an IP problem can be found from the LP version (or *LP-relaxation*) of the problem in which the integrality constraints on decision variables are relaxed. This is significant because LP problems can be solved much more quickly than IP problems. It can be shown that if the coefficient matrix of an IP problem is totally unimodular, the decision variables of its LP-relaxation always take on integer values [NeW99]. A matrix is *totally unimodular* if the determinant of each square submatrix of the coefficient matrix is equal to 0, 1, or -1. If the coefficient matrix is not totally unimodular, LP-relaxations can sometimes provide tight lower bounds on the optimal solution. Either way, IP (and LP) models are quite significant for practical problems and research use because of the availability of commercial optimization software. The formulation of planning problems as LP/IP models also provides a compact and precise way of representing and understanding the nature of many problems of interest.

2.6 Heuristics

In many real-world situations, mathematical programming techniques may require such a long time that they cannot be used in practice. For example, in large-scale network planning problems it may be necessary to solve some subproblems many thousands of times and it may not be practical to find optimal solutions for each instance of the subproblem. In these cases, a heuristic is often used to find an approximate solution. A *heuristic* is a method that usually finds a good solution quickly but doesn't necessarily guarantee that the solution found is optimal, or even feasible. In addition, many heuristics do not state how close a particular solution is to an optimal one. Most heuristics are problem-specific and are ineffective for other problems. A *meta-heuristic* is a method that controls or guides several simpler heuristics to produce a solution for a large-scale problem. In general, the basic structure of a meta-heuristic can usually be applied to several classes of problems by changing some of the underlying heuristics.

Heuristic methods can be classified into several broad categories. Two common heuristic approaches are the greedy constructive method and the local neighbourhood search. A *greedy constructive heuristic* (or greedy heuristic) is an iterative procedure that creates a solution one element at a time. At each iteration, the locally best choice of element is added to the solution. In general, making the locally best choice at each iteration does not guarantee that an optimum solution will be found. Consider, for example, the game of chess where the optimal solution is to capture your opponent's king. A simple heuristic is to capture as many pieces as possible. In this context, capturing a pawn is like winning a battle but such a move may cause you to lose the war. In fact, a common strategy is to sacrifice a pawn in order to capture the other player's king. This example, taken from [Hu82], is a clear case where a greedy heuristic is not optimal.

There are, however, other examples where the greedy approach is optimal. One example, is Prim's algorithm [Hu82], which produces a minimum weight spanning tree by adding at each iteration the edge of least weight that does not create a cycle. It can be shown that a greedy heuristic is optimal in those cases where the solution space is convex. Roughly speaking, a set is *convex* if all line segments joining pairs of its points are in the set. Most greedy heuristics, however, are based on human intuition and are not so easily analysed.

A *local neighbourhood search* begins with an initial solution and searches a defined neighbourhood for a better solution. A *neighbourhood* is a set of solutions that can be reached from the current solution by a simple operation called a *move*. Examples of a move include adding a new element to the solution or removing an existing element from the solution. A move may also involve swapping an object in the solution with one that isn't or changing the order of the objects within the solution (e.g., in a scheduling problem). If a better solution is found within the neighbourhood of the current solution, the process starts again at the new solution. This continues until no further improvement can be found. The final solution is a local optimum with respect to its neighbourhood but may not be a globally optimum solution.

One disadvantage of the local neighbourhood search is it may get "stuck" in a local optimum. Some alternatives for escaping a local optimum are to expand the local neighbourhood by considering more involved moves or to start the whole process from a different initial solution. Another alternative is to allow "uphill moves", which allow the search to escape a local optimum in the solution space. There has to be some restrictions on such moves otherwise the entire solution space may be explored. Two popular methods that adopt this approach are Simulated Annealing and Tabu Search. These methods are discussed in further detail in subsequent chapters.

2.7 Computational Complexity

When comparing both optimal and heuristic algorithms it is often useful to consider their computational complexity in terms of two critical resources: time and memory space. The *time complexity* of an algorithm measures the rate at which its running time grows as a function of the problem size. The "size" of a problem is generally the number of inputs processed. For example, the size of a TSP is typically measured by the number of cities in the tour. Although running time is usually the most important constraint, the memory space requirements (e.g. main memory and disk space) may also be of concern. The rate at which an algorithm's memory space grows as a function of the problem size is called its *space complexity*.

There are many factors that influence the running time (and memory requirements) of an algorithm. These include the speed of the computer, the programming language and the skill of the programmer. The input data can also have an impact on the running time. Consider, for example, the problem of sequentially searching a list of length n for a particular value x. In the *best case*, the value x is in the first position in the list and only one value is examined. In the *worst case*, the value x is in the last position in the list and all n values must be examined. If we run the search on many different problem instances, the average number of values examined will converge to $n/2$. We call this the *average case* performance of the algorithm.

Of primary concern when estimating the run-time performance is the number of basic operations performed. A *basic operation* is an operation whose cost does not depend on the value of the operands. For example, the addition of two integers is a basic operation. Complexity can be determined empirically or theoretically. Empirical analysis is used to predict the average-case behaviour of the algorithm by running it on a suitable number of problem instances and observing the running time. Theoretical analysis is frequently used to determine worst-case behaviour. The complexity of an algorithm is often expressed using a special notation, called *big-Oh notation*. This notation expresses an upper bound on the algorithm's running time within a multiplicative constant. For example, if the upper bound on an algorithm's worst-case running time is $f(n)$, then we would say that the time complexity of the algorithm is "in the set $O(f(n))$ in the worst case" [Ree98]. If the actual running time of an algorithm is denoted by $g(n)$ and $f(n)$ is some expression for the upper bound, then the complexity of the algorithm can be expressed mathematically as follows:

$$g(n) = O(f(n)) \tag{2.7}$$

if and only if there exists positive constants c and n_0 such that

$$|g(n)| \leq c \cdot f(n), \qquad \forall n > n_0 \tag{2.8}$$

Constant n_0 is the threshold value for the problem size beyond which equations (2.7) and (2.8) are valid. For example, if the time complexity of an algorithm is either experimentally or theoretically observed to quadruple when the problem size doubles, then we would say the complexity of the algorithm is $O(n^2)$. An algorithm is said to be *efficient* if the upper bound on running time $f(n)$ is a polynomial function of n. Such an algorithm is also called a *polynomial algorithm*. Prim's algorithm for the minimum weight spanning tree problem is an example of an efficient algorithm because the computing effort grows as a low-order polynomial of the problem size. For the TSP, however, the time complexity is an exponential function of the problem size, even for the best known algorithms. Such algorithms are called ***exponential algorithms***.

2.7.1 Classes P and NP

An important question in the development of any algorithm is whether the problem at hand can be solved in polynomial time. In complexity theory, a problem for which a polynomial time algorithm is known to exist is said to be in class *P*. The problems in class *P* also belong to a broader class of problems called *NP* problems. *NP* is an abbreviation for ***non-deterministic polynomial***. Technically, *NP* is the class of all problems that can be solved in polynomial time on a non-deterministic computer [GaJ79]. A non-deterministic computer is a hypothetical computer that can evaluate an unbounded number of "guesses", each of which can be checked in polynomial time. Strictly speaking, problems in class *NP* are decision problems only. The answer to a decision problem is either "yes" or "no". For example, the "decision version" of the TSP asks whether there is a tour of length *L*, rather than asking "what is the length of the optimal tour?" The later question is referred to as the "optimization" version of the problem. Note that we can convert an optimization problem to a decision problem and solve it using a binary search method.

A problem P_1 is ***reducible*** to problem P_2 if it can be transformed into P_2 in polynomial time. Because the class of instances of P_2 contains P_1, P_2 is at least as hard to solve as P_1. An ***NP-complete*** problem is a problem that is in *NP* and every other problem in *NP* can be reduced to it. If L_1 is *NP*-complete and L_1 can be reduced to L_2, then L_2 is also *NP*-complete. An optimization problem is ***NP-hard*** if its decision version is *NP*-complete.

An open question that has eluded computer scientists for over thirty years is whether $P = NP$. To date, the answer is still not known but it has been shown that if there is a polynomial algorithm for at least one *NP*-complete problem, then every *NP*-complete problem is solvable in polynomial time. Because all attempts to prove that $P = NP$ have failed and no exact polynomial algorithm has yet been found for any *NP*-complete problem, the evidence strongly suggests that no such algorithm exists. From a practical point of view, if a particular problem is known to be *NP*-complete (or *NP*-hard), the use of a heuristic algorithm is strongly justified because it is unlikely that an exact algorithm can be found for the problem. This practical implication is what motivates the use of heuristics in this thesis.

3 Transport Networks

3.1 Introduction

This chapter provides a tutorial introduction to transport network technology, concepts, terminology and other background information that are relevant to the research presented in this thesis. The chapter opens with a broad definition of a transport network and a classification of the various types of transport networking technology. Two of these technologies are then described in detail with emphasis on the most relevant attributes and concepts. The remainder of the chapter discusses survivable network architectures and, in particular, survivable ring architectures.

3.2 Transport Networking Technology

A *transport network* is a collection of high-capacity transmission, multiplexing and switching facilities that conveys a set of demands between nodes in a telecommunications network. A *demand* is defined here as the transmission capacity required between a pair of nodes in a network. Like Wu [Wu92], we use the general term *demand* rather than *traffic* because *traffic* is often associated with the volume of calls in a telephone network (e.g., measured in erlangs). A transport network may also transfer network control information used for operations, maintenance and provisioning purposes. The *nodes* in the network are usually located at telephone switching offices, commercial buildings or other locations where demands originate and terminate. In general, demands (and network control information) may be either analog or digital signals and may require either unidirectional, bidirectional or asymmetric transmission, depending on the type of service supported. Point-to-multipoint transmission may also be required for some types of services (e.g., broadcast video). Unless otherwise noted, all demands are assumed to be point-to-point and bidirectional throughout the remainder of this thesis.

Historically, transport networks were designed to carry telephone traffic between switching centres in the *Public Switched Telephone Network* (PSTN). Over the years, these networks have evolved to support a broad range of services (e.g., voice, data, video). These services may be served either directly or indirectly through service layer networks. Direct services are usually provisioned on a dedicated (or leased-line) basis and are typically used for applications such as broadcast video, high-speed computer-to-computer communications and private networks. Services that are supported indirectly through service layer networks include plain old telephone, cellular telephone, paging, video conferencing and data communications. Service layer networks differ from transport networks in that they are usually designed for a specific type of service and typically contain user signalling and control functions for setting up and tearing down calls or connections.

Examples of service layer networks include the PSTN, X.25, Frame Relay, the Internet and *Asynchronous Transfer Mode* (ATM). In contrast, transport networks provide bulk transmission and routing of individual and aggregate information streams independent of the type of end-user services supported. For this reason, they are sometimes referred to as "backbone" networks.

Conceptually, transport networks may be further divided into logical and physical layers [Wu92]. The *logical layer* is responsible for end-to-end routing of information streams and the physical layer consists of fibre optic cables, active and passive photonic devices, signal regenerators, terminal electronics and other equipment that provides the foundation for the higher network layers. Figure 3.1 shows a layered view of the service and transport networks. Note that this view should not be confused with the seven-layer reference model of Open Systems Interconnect (OSI) developed by the International Standards Organization (ISO) [BeG92].

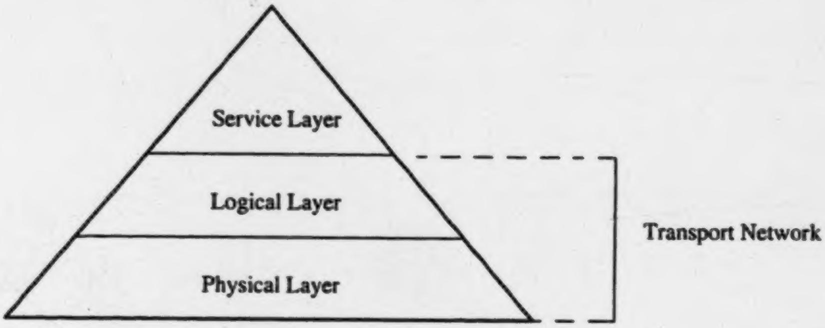


Figure 3.1. Layering of service and transport networks.

As traditional transport network and data communications technologies converge, the distinction between transport and service layer networks is blurring. For example, the Internet backbone network is now rivalling traditional transport technology in terms of switching capacity and transmission rates. Backbone *Internet Protocol* (IP) routers with Terabit/second switching fabrics and high-speed optical interfaces are now commercially available, making it possible to construct multi-service networks without traditional transport technology [Cis99]. Similarly, signalling and control functions have been proposed for transport networks to allow high data-rate connections to be established and dropped as demand dictates [Mac91].

Transport networking technology can also be classified according to the type of switching, multiplexing and transmission used within the network. As shown in Fig. 3.2, there are two basic types of switching that may be used in transport networks: packet-switching and circuit-switching.

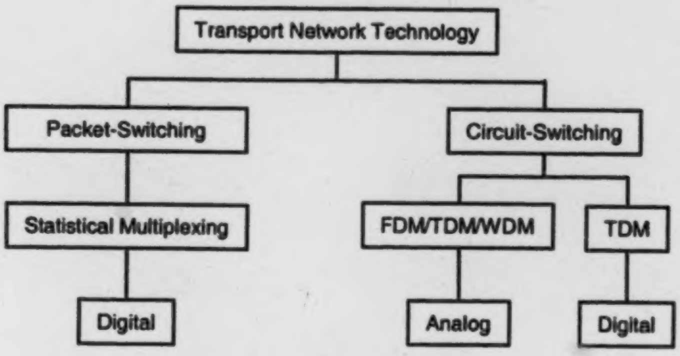


Figure 3.2. Classification of transport networks.

In a packet-switched network, the information flowing from one node to another is broken down into a sequence of packets (or *cells*) at the originating node before being transmitted over the network to the destination node(s). When a packet arrives at an intermediate node, it waits in a queue to be transmitted over the next transmission system enroute to its destination. At the destination node, the packets are reassembled to reconstruct the original information stream. Because the packets occupy the full capacity on a transmission system only while they are being transmitted, the transmission capacity can be shared over time with many other connections (or sessions). This is referred to as *statistical multiplexing* [BeG92]. Packet switching is particularly well suited for data sessions that are characterized by short bursts of high activity followed by long periods of inactivity. One disadvantage of packet switching is that queuing delays at the switching nodes are difficult to control. This can result in buffer overflow and loss of data.

The other basic type of switching that may be used in transport networks is circuit-switching. Circuit-switching is the primary method of switching used in traditional transport networks. For example, SONET and WDM optical networks (discussed later) are "circuit-switched" in the strict sense. In a circuit-switched network, each connection is allocated a given amount of transmission capacity (usually in both directions) between the origin and destination node. That is, on each transmission (and switching) facility along the path from the origin to the destination node, a portion of the capacity is dedicated to the connection. Therefore, once the path (or circuit) has been established, the connection has a guaranteed transmission capacity through the network for its entire duration. Unlike packet-switched networks, the capacity allocated to individual connections cannot be used by other connections during inactive periods. Furthermore, the sum of the capacity allocated to all paths on a given transmission facility cannot exceed its total capacity. Thus, if a transmission facility is fully allocated it cannot accommodate any new connections. If no other paths with the required capacity can be found through the network, a new connection request must be rejected or *blocked*. In contrast, an overload situation in a packet-switched networks results in an increase in queuing delays and potential loss of data due to buffer overflows.

Circuit-switched transport networks usually differ from circuit-switched (service-layer) networks such as the PSTN in several important ways. First, the duration (i.e., holding time) of a transport network connection is generally on the order months or years, as opposed to minutes for a typical PSTN call. For this reason, transport network connections are often referred to as "semi-permanent" or "nailed-up" connections. In addition, the data rate of transport network connections typically ranges from 1.5 Mbps up to 155 Mbps or more, whereas the data rate for a PSTN call is 64 kbps. In fact, because the term "circuit" is usually associated with a 64 kbps PSTN call, transport networks are sometimes referred to as "channel-switched" networks to distinguish them [Min91]. Lastly, PSTN circuit switches contain sophisticated call processing hardware and software, whereas switching facilities in transport networks are usually much more rudimentary. Often, they consist of nothing more than manual patch panels or, at best, electronic versions with simple cross connection management capabilities.

The most common forms of multiplexing used in circuit-switched transport networks are *frequency-division multiplexing* (FDM), *time-division multiplexing* (TDM) and *wavelength-division multiplexing* (WDM). In FDM, several message signals share the same physical channel by dividing the available channel bandwidth (in Hz) into non-overlapping frequency bands and assigning a separate band to each message signal. Analog FDM transmission systems were widely used in the PSTN until about the early 1980s. The North American FDM hierarchy (also known as the Bell System) used a modular FDM structure where up to 13,200 voice (or voiceband data) channels could be multiplexed for transmission over coaxial cable. Using a system of multiple tube pairs, the L5E carrier system (circa 1978) could carry up to 10 such FDM signals for a composite capacity of 132,000 voice channels [Min91, pp. 3]. Even today, the capacity this system is impressive.

Since the 1980s, analog FDM transmission systems have been largely replaced by digital TDM transmission systems using fibre optic technology. Fibre optics offers a number of advantages over other media such as metallic cables and microwave radio. These include higher capacity, higher reliability, longer repeater spacing, smaller size, less weight, immunity to electro-magnetic interference and lower system costs [Min91, pp. 334]. In a TDM system, several message signals are transmitted over the same channel by dividing the time frame into separate *time-slots*, one time-slot per message signal. In comparison with analog FDM systems, digital TDM systems require much simpler circuitry and are immune to crosstalk due to intermodulation distortion. They can also operate at lower signal-to-noise ratios because the digital signals can be regenerated at each repeater site, unlike analog signals, which accumulate noise along the entire path from origin to destination. Although digital signals are subject to other transmission impairments such as jitter and wander, digital transmission systems are much easier to design and maintain than their analog counterparts and, therefore, have become the predominant mode of transmission in telecommunications networks today. The two most common TDM standards for transport networking are the *plesiochronous digital hierarchy* (PDH) and the *synchronous optical network* (SONET*)* standard. These standards, particularly the SONET standard, are most relevant to this thesis and are described in detail in Sections 3.4 and 3.5.

WDM technology is now being deployed in transport networks to satisfy growing demand for transport capacity, due largely to the exponential growth in Internet traffic. In WDM systems, several optical carrier signals share the same fibre by dividing the optical bandwidth into non-overlapping channels and assigning each channel to a particular optical carrier signal. This is essentially the same as FDM, except at optical frequencies (i.e., wavelengths in the 1330 - 1550 nm range). For this reason, WDM is sometimes called optical frequency-division multiplexing [Kam93]. In the literature, the optical carrier signals are usually referred to as wavelengths (or λs). The main drivers for the deployment of WDM systems are the rapid depletion of fibre capacity in long-haul networks (due to demand growth) and the favourable economics of WDM systems relative to other alternatives, such as installing new fibre plant or replacing existing TDM systems with new higher data rate TDM systems.

Although the capacity of early point-to-point WDM systems was limited to 2 or 4 λs, systems capable of carrying up to 80 λs are now commercially available and systems with over 1,000 λs have been demonstrated. While the majority of WDM systems that have been deployed thus far are point-to-point WDM systems, optical ADMs (OADMs) and optical cross-connects (OXCs) are in field trials and the early stages of deployment. These network elements are now being used to create fully *optical networks* in which wavelengths are routed and switched in much the same way as time-slots are in SONET networks.

Today packet-switching is mostly used in service layer networks whose "logical bit pipe" connections are provided by circuit-switched transport networks such as SONET. With data now exceeding voice as the dominant form of traffic [Rya98], packet-switching is likely to play a more prominent role in future transport networks. For example, several proposals have been made to carry ATM or IP traffic directly over optical networks, thereby eliminating the need for an intermediate SONET layer [DPW99]. Although packet-switched optical network elements have been proposed [CCR99], evolving optical transport network standards are circuit-switched and closely resemble SONET network architectures. Indeed, there may always be a need or an advantage for circuit-like logical constructs in the transport layer. See [McB98] for a discussion of optical transport network architectures and emerging standards.

3.3 Generic Functional Architecture

Although different multiplexing and transmission technologies are used to implement the circuit-switched technologies described in the previous section, the basic routing and switching functions that they provide are essentially the same. The main difference is the basic unit used for measuring and allocating capacity. In SONET, for example, the basic unit of allocation is a timeslot or STS-n channel, whereas in optical networking it is a wavelength (or to be precise, an optical channel with a specified bandwidth). Of course, there are other important differences between these technologies. For example, SONET networking involves digital transmission engineering, whereas optical networking involves analog transmission engineering. But from a functional perspective, these networking technologies can be modelled in a generic fashion using the same abstract concepts [ITU95]. This has several implications. First, because the basic functions are equivalent in nature, the same types of network elements and architectures have been implemented across the range of technologies. For example, survivable ring architectures that where first implemented in SONET, are now being considered for the optical network layer [McB98]. Second, the design methods discussed in following sections of this thesis are broadly applicable across all network layers. For convenience, SONET terminology is used throughout the remainder of this thesis, unless otherwise noted.

In practice, transport networks are large and complex and usually consist of several different networking technologies. This is due in part to the evolutionary process through which new transport technologies are introduced and older ones are phased out. It also reflects the fact that different technologies may be required to best fulfil the wide range of service requirements. For planning and administrative reasons, it is usually convenient to decompose transport networks into separate hierarchical layers by the type of transport technology. Figure 3.3 illustrates this layered view of the transport network.

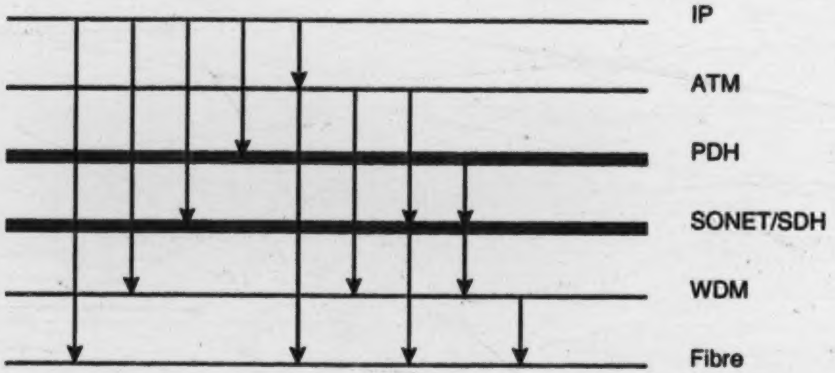


Figure 3.3. Layered view of a transport network (client/server association).

The adjacent layers in the network form client-server relationships, in which server layers perform signal multiplexing, transport and routing functions for one or more client layers. For example, the SONET layer can accept payloads directly from either the PDH, ATM or IP layers with the appropriate tributary interfaces. In turn, the resource requirements from the SONET layer become payloads for the WDM or fibre layers.

Each network layer can also be partitioned horizontally into tiers or subnetworks according to geographic and/or administrative boundaries, as illustrated in Figure 3.4. This further facilitates design and operation and allows for different survivability schemes to be implemented autonomously. In practice, this partitioning usually reflects the differences in demand distribution, cost structures and topological layout within a network layer. For example, it is common practice to partition a network layer into separate access, metropolitan inter-office (or metro) and core (or long-haul) subnetworks. In an access subnetwork, most demands originate at remote switching offices and customer premises and terminate at one or more main switching offices (or *hubs*). A metro subnetwork connects main switching offices (or other points of concentration) within a metropolitan area and demands are typically more uniformly distributed. Because span distances in access and metro subnetworks are typically less than 25 to 50 km, nodal equipment costs (e.g., ADMs, DCSs) usually dominate total network costs. Long-haul subnetworks, on the other hand, usually connect metropolitan areas on a national or international scale. In these networks, distance related costs such as fibre material and installation, amplifier and regenerator costs typically dominate the total cost because span distances are so much greater.

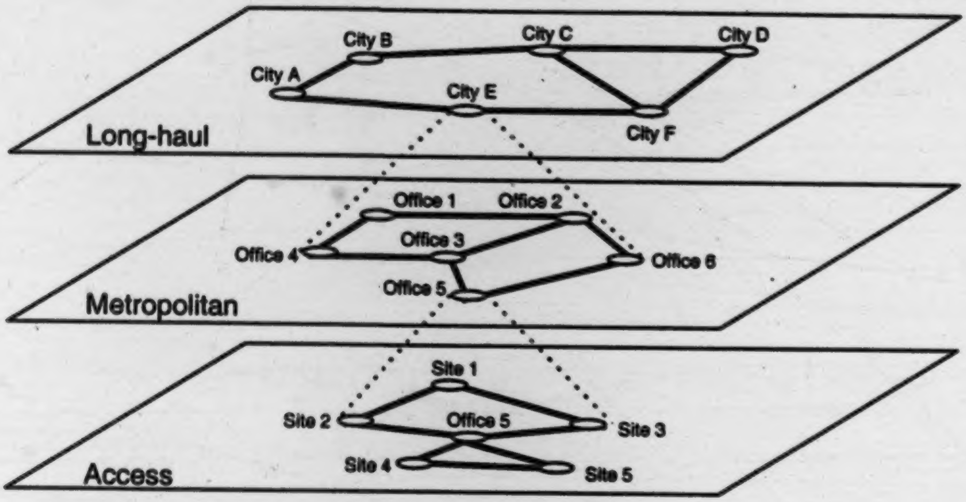


Figure 3.4. Partitioned view of a transport network.

Although the concepts of layering and partitioning are not the main focus of this thesis, it is important to mention them here because they simplify the design process. That is, it is simpler to design (and operate) each subnetwork within a layer than to design the entire network as a single entity. The multi-layer design is an important area of research but is beyond the scope of this thesis. For a discussion of multi-layer design issues see [Dem99].

3.4 Plesiochronous Digital Hierarchy

The first TDM system to achieve widespread use was the T-1 carrier system, developed at Bell Labs in 1962 [Min91]. Initially, these systems were deployed in metropolitan areas to support telephone trunks between switching offices. T-1 carrier systems use *pulse code modulation* (PCM) to convert speech from analog to digital form. In this process, the voice signal is first passed through a 3.4 kHz lowpass filter and sampled at 8 kHz. Then each sample is quantized into one of 256 levels using a logarithmic scale and encoded into an 8-bit codeword. The resulting 64 kbps digital signal is called a Digital Signal-0 (DS0) in the *plesiochronous digital hierarchy* (PDH). A T-1 multiplexer (or channel bank) combines twenty-four DS0s by byte-interleaving the 8-bit codewords from all 24 voice channels every 125 μseconds. A single framing bit is also added to each *frame* of 24 coded voice signals (192 bits). The combined transmission rate of the aggregate signal is 193 bits every 125 μseconds or 1.544 Mbps. This signal is called a Digital Signal-1 (DS1).

A similar PCM transmission system, known as E-1 carrier, was also developed in Europe. In an E-1 carrier system, the basic signal rate is also 64 kbps but a different method is used to convert each voice sample to an 8-bit codeword. Thirty voice channels (i.e., E0s) are multiplexed by byte-interleaving the 8-bit samples from each channel and an additional 16 bits of overhead are then added for framing and signalling. Because the duration of each 256-bit frame is 125 μseconds (i.e., 8,000 samples per second), the data rate of the aggregate "E-1" signal is 2.048 Mbps.

As higher digital transmission rates became possible, a set of standard signal rates were developed in North America and Europe for digital signal multiplexing. Table 3.1 lists the standard signal rates and multiplexing ratios for the North American PDH. The multiplexing ratio defines the number of lower-speed *tributary* signals that can be multiplexed into each digital signal. For example, a DS2 signal is constructed by byte-interleaving four DS1 tributary signals and adding some additional overhead bits. Seven DS2 signals are then multiplexed to form a DS3 signal and so on. Initially, these signals were carried over twisted copper pairs, coaxial cable and microwave radio. Interfaces for early fibre optic transmission systems were developed later.

Table 3.1: North American Plesiochronous Digital Hierarchy

Signal Level	Data Rate (Mbps)	Composition
DS0	0.064	-
DS1	1.544	24 DS0s
DS2	6.312	4 DS1s
DS3	44.736	7 DS2s
DS4	274.176	6 DS3s
DS5	560.160	2 DS4s

In both PDH hierarchies, *pulse-stuffing* is used at the second multiplexing stage (e.g., DS2) and higher, to accommodate differences in the clock frequency and phase of tributary signals. Pulse-stuffing works by inserting additional bits (some fixed, some inserted adaptively to match average payload clock to the plesiochronous carrier signal clock) into the aggregate signal to meet its nominal signal rate. These bits are then removed at the far end when demultiplexing the tributary signals. Because the pulse-stuffing mechanism can tolerate only slight deviations from the nominal tributary rate, the clock frequency of these tributaries are *plesiochronous*, which means "almost synchronous." The term *asynchronous* is also used to describe both digital hierarchies but is less precise. One disadvantage of pulse-stuffing is that the aggregate signal must be completely demultiplexed to access (i.e., add or drop) individual tributary signals. For example, to drop a DS1 from a DS3, the entire DS3 signal must undergo two stages of demultiplexing, first to the DS2 rate and then to the DS1 rate. Once the desired signal has been dropped, the remaining DS1s must be re-multiplexed back up to the DS3 level. This adds to the complexity and cost of multiplexing equipment. Another limitation is the lack of a uniform set of functions for network performance monitoring, fault detection, provisioning and other network management features and capabilities. These limitations were among the motivating factors for the development of the new SONET standard, discussed in the next section.

3.5 Synchronous Optical Network (SONET)

The *Synchronous Optical Network* (SONET) standard was developed in the late 1980s in North America to address limitations of earlier PDH systems, realize higher data rates possible with fibre optic transmission media and provide a standard optical interface signal specification to facilitate interconnection of equipment from different vendors [Min91]. The international equivalent of SONET is the *Synchronous Digital Hierarchy* (SDH), which is specified by the ITU-T [ITU93]. For simplicity, SONET terminology is used in the remainder of this thesis.

The basic building block in the SONET signal hierarchy is the *synchronous transport signal - level 1* (STS-1) [ANS95a]. The STS-1 frame is usually depicted as a matrix of nine rows by 90 columns of 8-bit bytes, as shown in Fig. 3.5. The bytes are transmitted in order from left to right beginning with the first row. The entire frame is transmitted in 125 μs. With a total of 810 bytes (6480 bits) and a frame duration of 125 μs, the STS-1 data rate is 51.840 Mbps. The STS-1 frame is divided into two portions: transport overhead and a *synchronous payload envelope* (SPE). Transport overhead occupies the first three columns of the frame and is further divided into line and section overheads, which provide signal framing, line identification, performance monitoring, and voice and data channels (used for provisioning and maintenance). The SPE occupies the remaining 87 columns by nine rows (783 bytes). The first column (9 bytes) of the SPE is used for path overhead functions such as end-to-end performance monitoring and path identification. The other 86 columns (6192 bits) are allocated to the payload signal(s). The maximum data rate of the payload signal(s) is 49.536 Mbps.

Unlike earlier PDH systems that use pulse stuffing for synchronization, SONET uses a pointer mechanism to identify the start of the SPE within the STS-1 frame. The STS-1 pointer is contained in the transport overhead and indicates the offset between the position of the transport overhead and the SPE. This approach provides the ability to adjust for slight variations in the frequency of

STS-1 payloads and to access the STS-1 payload directly from a higher rate signal without demultiplexing the entire signal. A similar pointer mechanism is used to access virtual tributaries (discussed below) within an STS-1 SPE. Single-step multiplexing reduces the number of components in an end-to-end path, which improves reliability and reduces the complexity and cost of multiplexing equipment. It also allows efficient and cost effective implementation of survivable ring architectures, which are discussed in detail in Section 3.7.

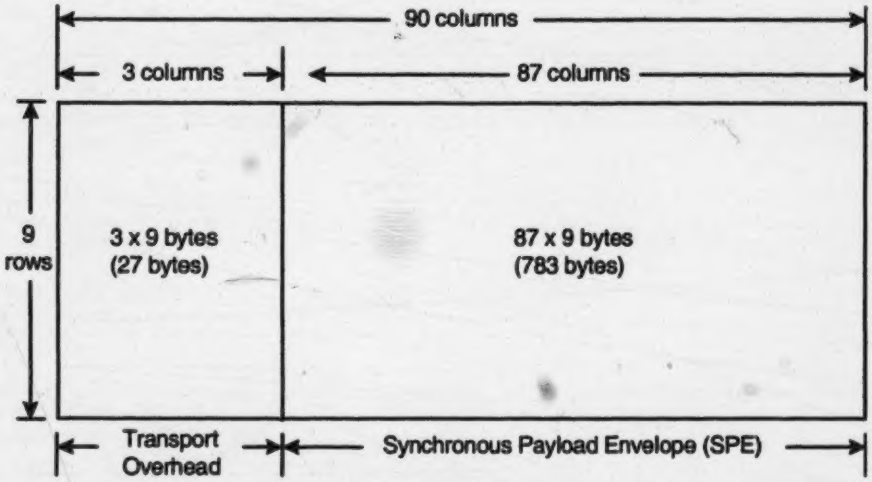


Figure 3.5. SONET STS-1 frame.

An STS-1 SPE can be used to carry a single DS3 (44.736 Mbps) or it may be subdivided into smaller envelopes to provide backwards compatibility with lower bit rate PDH signals. For example, a DS1 signal can be carried within an STS-1 SPE by mapping it into a SONET unit called a *virtual tributary*-1.5 (VT1.5). An STS-1 SPE can carry up to 28 VT1.5s. Several VT mappings have also been defined for other sub-DS3 signals.

Higher rate SONET signals are obtained by byte-interleaving a whole number of STS-1s. Services that require multiples of the STS-1 rate (e.g., ATM) can be transported as a unit by concatenating several STS-1 signals together. Higher rate signals can be comprised of any combination of lower rate individual STS-1s or concatenated STS-Nc signals. For example, an STS-12 signal can be created from 12 STS-1 signals or 4 STS-3c signals or any other combination of STS-1 and STS-3c signals that equals STS-12. Prior to transmission, the STS-N signal is scrambled and converted to a corresponding *optical carrier signal* (OC-N) via electrical-to-optical conversion.

Table 3.2 lists the most common SONET signal levels and their data rates.

Table 3.2: SONET Digital Signal Hierarchy

Signal Level	Optical Signal	Data Rate (Mbps)
STS-1	OC-1	51.84
STS-3	OC-3	155.25
STS-12	OC-12	622.08
STS-24	OC-24	1244.16
STS-48	OC-48	2488.32
STS-192	OC-192	9953.28

As shown in Figure 3.6, SONET overhead and transport functions are divided into three layers: section, line and path. The section overhead provides framing and performance monitoring for the STS-n signal and local voice and data communications channels. Network equipment that terminates the section overhead is called *section terminating equipment*. This includes regenerators, terminal multiplexers, add/drop multiplexers and digital cross-connect systems.

Regenerators are used at intermediate points along the line to extend the transmission distance. This is done by converting the optical signal to the electrical domain, retiming and reshaping the STS-N signal, and then retransmitting it in the optical domain. Optical amplifiers may also be used between regenerators to increase the power level of the optical signal without *optical-to-electrical* (O/E) conversion and further extend transmission distance. Terminal multiplexers, add/drop multiplexers and digital cross-connect systems are described in detail below.

The line overhead provides performance monitoring of individual STS-1s, SPE pointer adjustment, and voice and data communications channels for OAM&P. Network equipment that terminates the line overhead is called *line terminating equipment*. The path overhead is transported along with the SPE until it is demultiplexed. A SONET *path* is a network connection at a given data rate between the point where the SPE is assembled and the point where it is disassembled. SONET equipment that originates/terminates the SPE and the path overhead is called *path terminating equipment*. Line and path terminating equipment can be any SONET equipment except a regenerator. The three basic types of SONET equipment are described in the following sections.

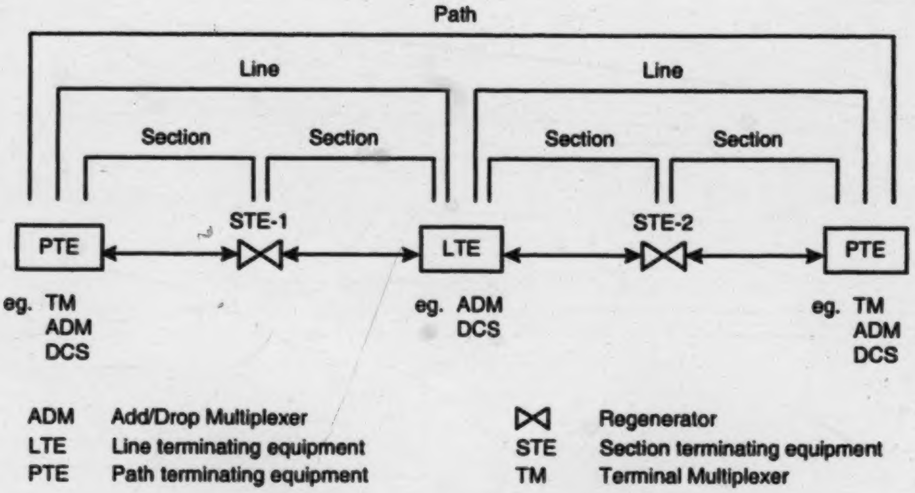

Path
Line
Line
Section
Section
Section
Section
STE-1
STE-2
PTE
LTE
PTE
eg. TM
ADM
DCS
eg. ADM
DCS
eg. TM
ADM
DCS
ADM Add/Drop Multiplexer
LTE Line terminating equipment
PTE Path terminating equipment
⋈ Regenerator
STE Section terminating equipment
TM Terminal Multiplexer

Figure 3.6. SONET section, line and path equipment.

3.5.1 *Terminal Multiplexer*

A *terminal multiplexer* (TM) is a device that terminates several tributary signals (e.g., DS1, DS3), assembles them into one or more SPEs, and converts the resultant electrical STS-N signal into an optical carrier OC-N signal for transmission. A functional block diagram of a TM is shown in Figure 3.7.

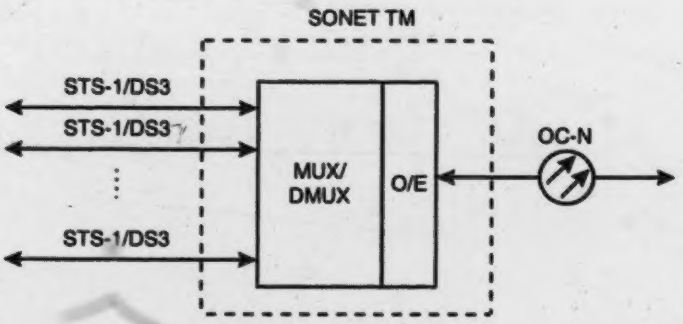

SONET TM
STS-1/DS3
STS-1/DS3
STS-1/DS3
MUX/
DMUX
O/E
OC-N

Figure 3.7. Functional block diagram of a TM.

Most TMs support a range of electrical and optical tributary types and either single or protected (e.g., 1+1) line interfaces (e.g., OC-3, OC-12, etc.). Terminal multiplexers may be used in traditional point-to-point systems or as part of linear add/drop chains, as depicted in Figure 3.6.

Because these multiplexers terminate the entire optical line signal, they are sometimes referred to as *line terminating equipment* (LTE).

3.5.2 Add/Drop Multiplexer

An *Add/drop multiplexer* (ADM) is similar to a TM except there are two line interfaces. A functional block diagram of an ADM is shown in Fig. 3.8. The line (or *high-speed*) interfaces are usually called the East and West line interfaces. ADMs are used at intermediate sites along linear add/drop chains (like the one depicted in Figure 3.6) to allow tributary signals to be added or dropped from the line signal or to pass-through enroute to their final destinations. ADMs may also be used in survivable ring architectures, which are discussed in detail in Section 3.7. Like TMs, most ADMs also accept a variety of electrical and optical tributary (or *low-speed*) interfaces.

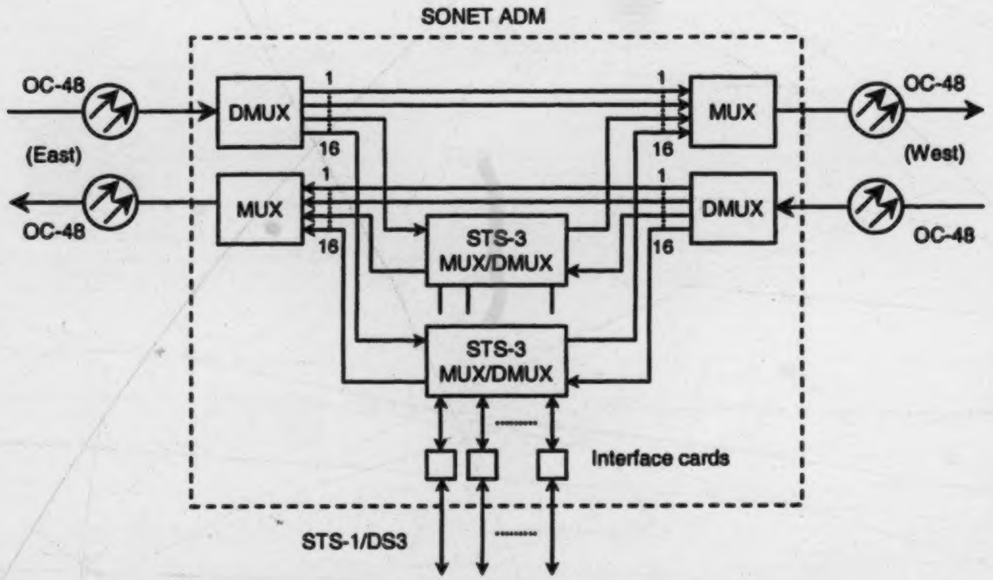


Figure 3.8. Functional block diagram of an ADM (adapted from [Wu92]).

In a SONET ADM, signals that pass-through the site do not need to be demultiplexed and then multiplexed back into the line signal. In comparison with back-to-back TM arrangements, this provides cost savings and improved reliability. ADMs are sometimes equipped with time-slot interchange capability to allow flexible assignment of any tributary signal to any east bound or west bound SPE. The implications of this feature in survivable rings architectures are discussed in Chapter 4.

3.5.3 *Digital Cross-Connect System*

A *digital cross-connect system* (DCS) is a SONET network element that accepts various electrical and optical carrier signals, accesses the individual tributary signals (e.g., VTs, STS-1, STS-Nc) and switches them from incoming to outgoing facilities. In addition to this switching function, most DCSs also provide signal add/drop and multiplexing/demultiplexing. There are two common types of DCSs: broadband DCSs and wideband DCSs. A *broadband DCS* (B-DCS) accepts OC-N and DS3 signals and cross-connects them internally at the DS3, STS-1 and/or STS-Nc rates. A functional block diagram of a B-DCS is shown in Figure 3.9.

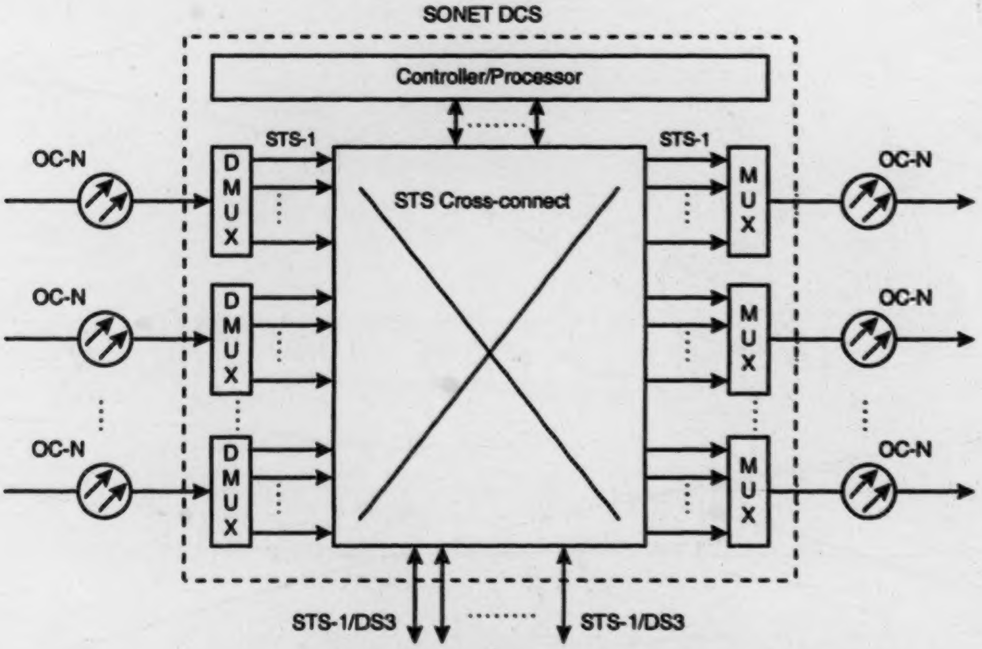


Figure 3.9. Functional block diagram of a B-DCS (adapted from [Wu92]).

A *wideband DCS* (W-DCS) is similar to a B-DCS except that switching is done at the DS1 or VT-1.5 rate. One advantage of W-DCS is that less demultiplexing is required because only the required tributaries are accessed and switched. A major difference between a DCS and an ADM, which has only two high-speed interfaces, is that a DCS usually has much more capacity. For example, a typical B-DCS can cross-connect up to 2048 DS3/STS-1 (or equivalent) ports. One of the main uses of DCSs in transport networks is ***signal grooming***. Grooming allows efficient use of incoming and outgoing facilities by consolidating demands into outgoing trunks and segregating demand by service type, destination or protection category. Consolidating demands improves the utlization of transmission facilities by combining tributaries from partially-filled incoming facilities into a smaller number of outgoing facilities. Segregation is also useful for simplifying maintenance and restoral procedures [Bell93]. One of the major advantages of DCSs, compared with manual cross-connect frames, is remote provisioning, maintenance and performance monitoring. This is made possible by the integrated multiplexing and cross-connect functions and SONET's embedded data communications channels, which allow DCSs and other network elements to be controlled remotely.

In addition to providing access for testing and maintenance, DCSs can also be used to provide service restoration by re-routing end-to-end paths onto spare facilities in the event of a failure. This is the basis for *mesh restoration*, which is discussed in the next section. Unlike PSTN switches, DCSs of any type are required to be fully non-blocking (to non-multicast demands) and consequently are often built upon multi-stage (3, 5, 7 stage) Clos non-blocking switch design principles.

3.6 Network Survivability

The survivability of transport networks to faults, such as cable cuts and equipment failures, is an increasingly important issue for customers and service providers alike. The need for enhanced network survivability in today's transport networks is well documented [Gro94]. Some of the key drivers for enhanced network survivability include increased reliance on telecommunications services, rapid growth in demand, higher transmission capacity and a greater concentration of network traffic onto fewer fibre spans.

The term *network survivability* is defined as the general (often qualitative) assessment of the capability of a network to continue to provide service in the event of network-related failures. Several metrics may, however, be used to quantify network survivability [Bel93]. One measure of network survivability is *conditional survivability*, which measures the fraction of services that continue to satisfy their *quality of service* (QoS) objectives for a specified type of network failure(s). Because the likelihood of multiple failures is typically quite low, conditional survivability is usually evaluated for all single span and node failures only. The conditional survivability of a single span failure is known as the *restorability* of a span [Gro94]. The restorability of a network is the total fraction of services that are protected over all single span failures. The *worst case survivability* measures the lowest fraction of services that survive all possible single-point failures. For example, a transport network that can sustain any single-point failure without the loss of service would have a worst case survivability of 100%. Unless otherwise noted, the term survivability will refer to a worst case survivability of 100% throughout the remainder of this thesis.

As shown in Figure 3.10, there are two generic strategies for making any network layer survivable: protection and mesh restoration. Protection techniques use preassigned spare facilities to protect against different possible failures. Protection schemes provide excellent survivability but require a large increase in network capacity relative to an unprotected network. Two protection schemes are *automatic protection switching* (APS) and survivable rings [Wu92]. APS is the simplest protection mechanism. In a 1+1 APS system, each working fibre is protected by a dedicated protection fibre. Tributary signals are bridged at the transmitting end and the receiver selects the better of the two signals. Equipment failures are protected against by using duplicate transmitter/receiver pairs for each working system. Span failures (e.g., cable cuts) are protected against by diversely routing the protection and working fibres through the duct topology. A 1:1 APS system is similar to 1+1 APS, except the transmitted signal is not continuously bridged to the protection facility. Instead, switching is performed at both the transmitting and receiving end in the event of a failure. There are also 1:N APS systems that protect N working fibre systems with a single protection fibre but they do not provide 100% survivability against span failures. APS has been adopted in the SONET standard as one of a few basic methods of facility protection [ANS95b]. According to this standard, protecting switching must be completed within 50 msec. after detecting signal failure. Transport networks using APS protection have a worst case survivability of 100% but require twice the transmission capacity (i.e., 100% redundancy) relative to a non-survivable design.

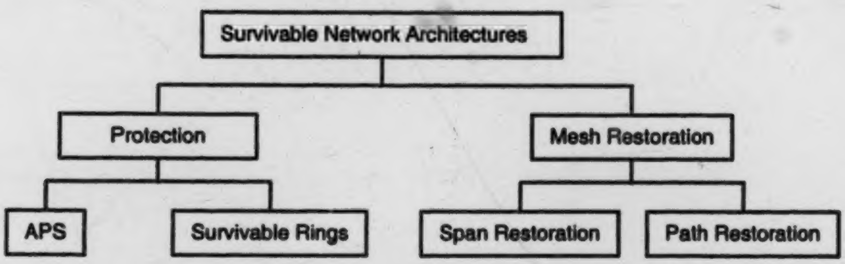


Figure 3.10. Classification of survivable network architectures.

Although traditional APS operates at the physical fibre layer, the same basic principle of dual feeding can be applied at other logical layers. For example, duplicate, physically diverse paths can be provisioned in the SONET layer for the same demand to provide end-to-end path protection. This is sometimes referred to as *subnetwork connection protection* (SNCP). Similar schemes have also been proposed for survivability at the ATM and IP layers [Wu92].

Survivable ring architectures can be viewed as an extension of 1+1 or 1:N APS systems. In a *survivable ring*, the nodes belonging to the ring are connected by working and protection fibres (or fibre pairs) to form a closed loop or cycle. In a *path-switched* ring, each demand is transmitted in opposite directions around the ring on both the working and protection fibres and, like 1+1 APS, the receiving node selects the better of the two signals. A *line-switched* ring protects demands by looping the working line signal back onto the protection fibre pair at the nodes adjacent to a failure. This is analogous to 1:N APS because the protection facility must be coordinated at both ends of the failure. Unlike 1:N APS, however, line-switched rings are 100% restorable. Although the protection switching speed of survivable rings is comparable to APS, they can be more economical because the facilities (e.g., ADMs, regenerators) are shared amongst several origin-destination (O-D) pairs. Survivable ring architectures are discussed in further detail in Section 3.7.

In *mesh restoration*, survivability is provided via the switching capabilities of DCSs in combination with spare network capacity. During failures, DCSs reroute disrupted demands around failures by temporarily forming paths over routes in the set of spare (or idle) capacity resources within the network. There are two common types of mesh restoration: span restoration and path restoration. In *span restoration*, the replacement paths are found for all disrupted demands between the end nodes of the failed span. In *path restoration*, the replacement paths are found between the origin and destination nodes of the disrupted demands. The main advantage of mesh restoration is that the spare capacity resources can be used to recover from several (non-simultaneous) failures. Although line-switched rings also share protection (or spare) capacity, the extent and generality of the sharing is greater in mesh restoration. As result, mesh restoration is usually much more capacity-efficient than APS or survivable rings. This efficiency, however, usually comes at the cost of increased complexity (e.g., managing the reconfiguration of spare channels into restoration paths) and historically slower restoration speeds (in the range of seconds to minutes). New mesh restoration schemes are, however, currently under development that promise restoration speeds comparable to physical layer alternatives [StG00]. Although several proprietary mesh restoration have been deployed, there are currently no standards for mesh restoration. See [Gro94] for a survey of centralized and distributed mesh restoration schemes.

While mesh restorable networks achieve the lowest redundancy in transmission capacity needed for 100% restorability, ring architectures are often preferred in practice because of their simpler and faster switching mechanism (50 ~150 milliseconds). Despite their greater capacity requirements, rings can also be more economical than mesh networks, particularly in metropolitan area networks, where nodal costs usually dominate over distance-dependent costs for fibre and regenerators. For these reasons, SONET-based rings have already been widely deployed and the same logical architectures are promising and obvious candidates for optical networks.

An important issue in the design of survivable transport networks is deciding which layer or layers within the network should provide survivability against network failures. Recovery at the lowest layer (e.g., SONET, WDM layer) offers a number of benefits. It provides universal protection for all higher client layers against certain types of failures such as cable cuts, which are the most frequent source of failure. This may be especially important for those client layers without any built-in restoration mechanisms of their own. It also requires less equipment and fewer actions to effect recovery at the lowest layer because the aggregate switching granularity is much coarser than in client layers and fewer logical connections are affected and need to be rerouted.

To recover from node losses within the respective client layers, however, some form of recovery must be provided at those layers as well. In the PSTN, for example, survivability can be provided by multi-hosting, multi-homing or dynamic routing [Min91, WCY99]. A multi-hosting configuration splits the traffic originating from a local telephone office and routes it to two different central telephone offices. In the event of a single link or central office failure, this configuration maintains partial communications to and from the local telephone office. Blocking levels increase but service is not disconnected. A multi-homing configuration is similar to multi-hosting except rather than splitting the traffic, a multi-homing configuration simply switches the traffic to a backup central office during a failure. Dynamic routing is also used within the PSTN to provide service survivability by automatically routing call traffic around failed spans and nodes. One disadvantage of relying solely on multi-homing, multi-hosting, and dynamic routing for restoration is that calls that are in progress when a failure occurs will be dropped and only new calls will be successfully rerouted. This can result in exceptionally high transient levels of blocking for some time after a failure, as callers attempt to re-establish dropped connections [Gro94]. Some service layer networks recover from node or span failures by dynamically updating the routing tables stored in switching nodes. In the Internet, for example, backbone routers (or packet switches) update their routing tables using the Open Shortest Path First (OSPF) protocol [Moy98].

The planning and coordination of recovery processes in different layers is an important issue but is beyond the scope of this thesis. For an introduction to multi-layer recovery issues see [Dem99].

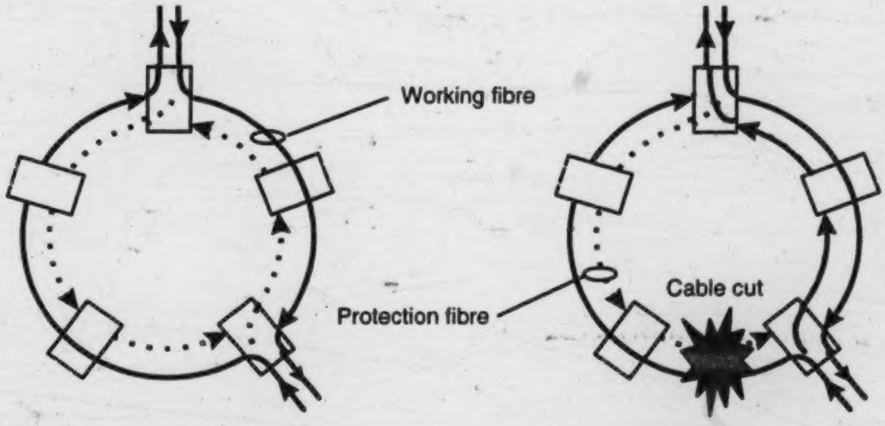
3.7 Survivable Ring Architectures

A *survivable ring* is a collection of nodes connected by transmission systems to form a cycle in the network graph. At each node, tributary signals may be added (multiplexed) or dropped (demultiplexed) from the composite line signal via ADMs. Tributary signals enroute to other nodes may also pass through an ADM, usually bypassing the demultiplexing and multiplexing stages. A ring may also contain passive (or *glassthrough*) nodes through which the transmission facility (e.g. fibre optic cable) passes but does not terminate on an ADM. Logical glassthrough nodes are often equipped with a signal amplifier or regenerator to meet optical link budget requirements. In this case, they are sometimes called "passthrough" nodes. Because there are two node-disjoint (and span-disjoint) routes between every pair of nodes on a ring, demands are protected against all single-point failures such as cable cuts or nodes failures.

There are two types of ring defined in the SONET standards. These are the *unidirectional path-switched ring* (UPSR) and the *bidirectional line-switched ring* (BLSR). These ring architectures are described in further detail below.

3.7.1 *Unidirectional Path-Switched Ring*

In a UPSR, the nodes are connected by a working and protection fibre, each of which transmits the line signal in the opposite direction. Figure 3.11 illustrates the basic operation of a UPSR.



(a) Normal Operation (before failure)

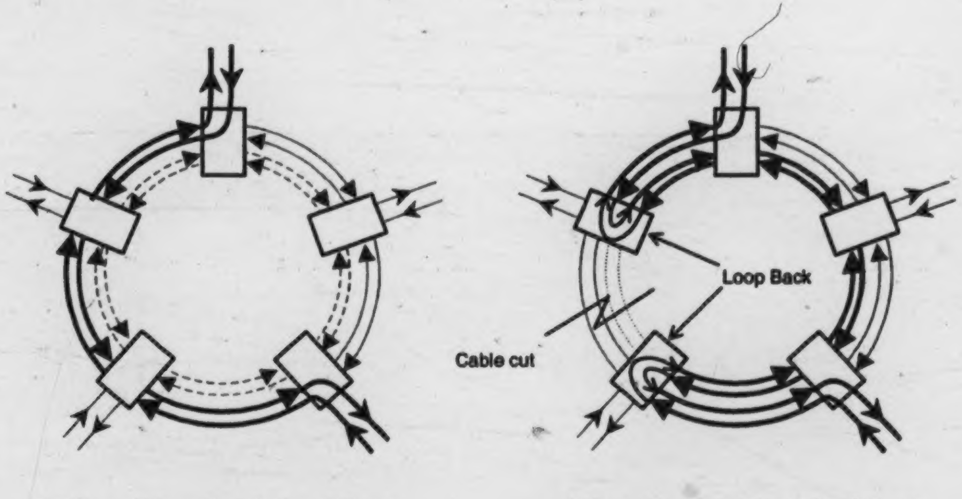
b) Protection Operation (after failure)

Figure 3.11. Two-fibre UPSR protection switching operation.

Under normal conditions, the demand between pairs of nodes in the ring is transmitted on the working fibre in one direction around the ring. A copy of each demand is also transmitted on the protection fibre in the opposite direction. At the receiving node, a path selector continuously monitors both signals and switches from the working to the protection fibre when the working signal is lost or degraded. Note that protection switching decisions are made individually for each path rather than for the entire line. According to the SONET standard [Bel95a] for UPSRs, the protection switching time for a UPSR must be less than 50 milliseconds after detection of signal loss or degradation. Because each bidirectional demand carried on the working (and protection) fibre traverses the entire ring, the total demand carried cannot exceed the number of channels (i.e., time-slots) in the ring. In other words, the capacity of the working (and protection) fibre must be equal to or greater than the sum of all demands carried by the ring.

3.7.2 Bidirectional Line-Switched Ring

In a four-fibre BLSR, the nodes are connected by a working and a protection fibre pair, as depicted in Figure 3.12. Unlike a UPSR, a bidirectional demand does not traverse the entire ring, instead it is routed over the same (typically shortest) path between the origin and destination nodes on the working fibre pair.



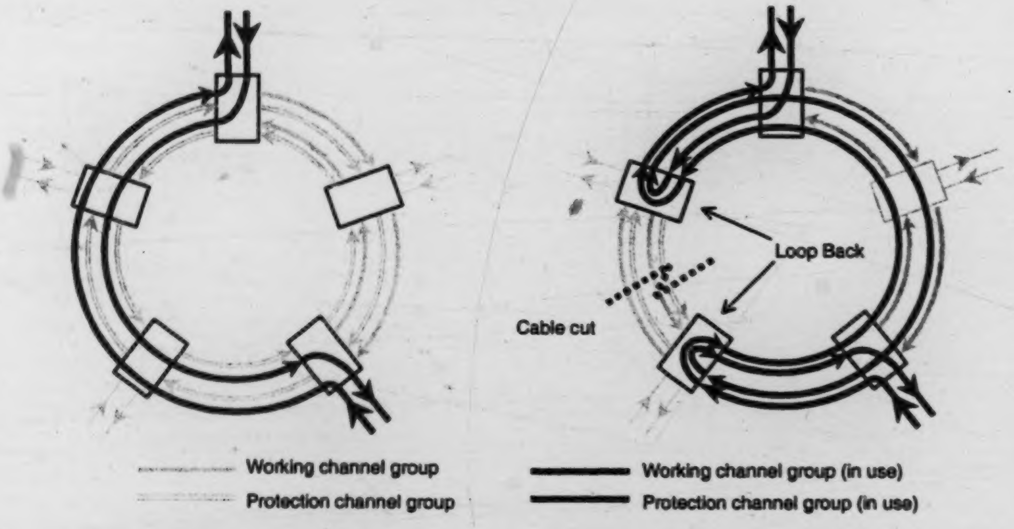
(a) Normal Operation (before failure)

(b) Protection Operation (after failure)

Figure 3.12. Four-fibre BLSR protection switching operation.

In the event of a cable cut or node failure, a BLSR restores working demands by looping them back onto the protection fibres at the nodes adjacent to the failed segment, which may contain one or more spans or nodes. Since protection switching is performed at both nodes adjacent to a failed segment, communications is required between these nodes to coordinate the protection switch. The two-byte *Automatic Protection Switching* (APS) message channel (bytes K1 and K2) in the SONET line overhead channel performs this function. Because the protection fibre may pass through one or more intermediate nodes before reaching its destination, addressing is also required to ensure that the APS message is recognized by the proper node and protection switching is initiated at the right pair of nodes. For this purpose, the SONET BLSR standard reserves four bits in the K1 byte for the destination node's ID and four bits in the K2 byte for the originating node's ID. Thus, the maximum number of nodes in a BLSR is limited to 16. According to the SONET standard for BLSRs [Bel95b], protection switching must be completed within 150 milliseconds after detecting signal loss or degradation.

A two-fibre BLSR, operates in logically the same way except the capacity on each fibre is divided into working and protection channel groups, as illustrated in Fig. 3.13. In the event of a failure, the working channel group is looped back onto the protection channel group at each node adjacent to the failure. In an OC-12 system, for example, a loop back is accomplished by mapping working STS-1 time slots 1 through 6 to protection STS-1 time slots 7 through 12 on the reverse direction fibre. The time slots for working demands at intermediate nodes are not affected by the fault.


Loop Back
Cable cut
Working channel group
Protection channel group
Working channel group (in use)
Protection channel group (in use)

(a) Normal Operation (before failure)

(b) Protection Operation (after failure)

Figure 3.13. Two-fibre BLSR protection switching operation.

One advantage of BLSRs over UPSRs is that the working channels (time-slots) can be reused around the ring. That is, because a demand occupies an STS-1 time-slot only between its respective entry and exit nodes, the same time-slot can be reused for other demands once it reaches its destination. Therefore, depending on the demand pattern and the routing choices for each demand pair, a BLSR can sometimes carry significantly more demand than a UPSR with the same working capacity. Another advantage of BLSRs is that the protection capacity is shared amongst all working fibre spans belonging to the ring. For this reason, BLSRs are sometimes called *shared protection rings* [Fla90]. Note that the protection capacity must equal or exceed the maximum working load on any span to ensure complete protection. Because the capacity (line rate) of a BLSR is the same on all spans, the required capacity z_r of both the working and protection fibres (or fibre in the case of a two-fibre BLSR) must be

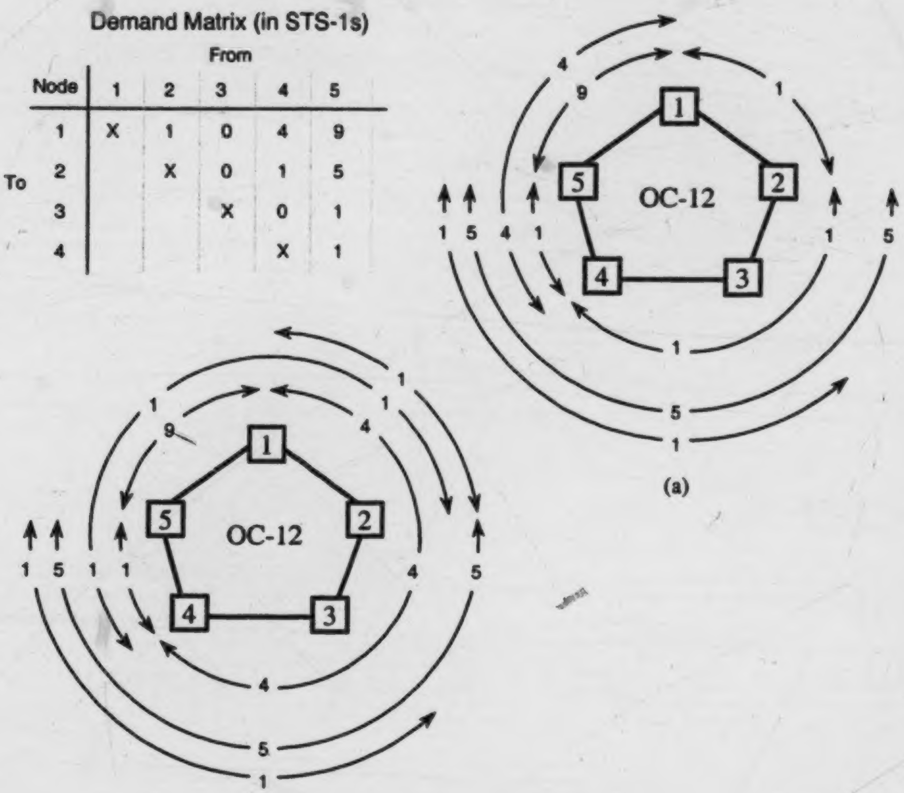
$$z_r \geq max(w_i), \qquad \text{for } i = 1,\ldots,n, \qquad (3.1)$$

where w_i is the working load on span i. Or, in other words, the working load on any span cannot exceed the working capacity of the ring. Note that the total capacity (working + protection) per span is $2 \cdot z_r$. In a four-fibre BLSR, the working (and protection) capacity is the same as the line rate. In a two-fibre BLSR, the working (and protection) capacity is half the line rate because half of the capacity is reserved for protection. For example, a four-fibre OC-12 BLSR has a working capacity of 12 working STS-1s per span, whereas an two-fibre OC-12 BLSR has a capacity of only 6 STS-1s per span.

Because the working load on any span is equal to the sum of all demands that traverse it, the required capacity (or demand serving capability) is dependent on the routing choices for all demands. This can be illustrated with the aid of Figure 3.14, which shows a demand matrix and two possible BLSR loading plans. In Figure 3.14(a), each demand is routed over the shortest path between the origin and destination. This results in a maximum working load of 13 STS-1s on span 1-5 (e.g., 4+9=13 STS-1s). Therefore, a BLSR with a working capacity of at least 13 STS-1s is required to serve all demands. Because SONET BLSRs are only available in standard sizes (typically in multiples of OC-12 x n), either a four-fibre OC-24 BLSR or a two-fibre OC-48 BLSR would likely be required for this routing arrangement. In Figure 3.14(b), the demands are first sorted in descending order of size and then routed over the path that results in the smallest change in required ring capacity. This routing procedure results a maximum load of only 10 STS-1s. In this case, either a two-four fibre OC-12 BLSR or a two-fibre OC-24 BLSR would be sufficient to serve all demands. This example clearly shows that demand routing has a direct impact on the either the size of BLSR required to serve all demands or the subset of available demands that can be carried on a BLSR of fixed size. These two subproblems are subject of Chapter 4.

Demand Matrix (in STS-1s)

To \ From: Node	1	2	3	4	5
1	X	1	0	4	9
2		X	0	1	5
3			X	0	1
4				X	1



(a)

(b)

Figure 3.14. An example of BLSR ring loading.

3.8 Summary

In this chapter we provided an overview of the basic concepts and terminology of transport networks. We also described the generic functions of transport networks and discussed two of the more common digital signal hierarchies (i.e., PDH and SONET) in use today. The typical network elements (or equipment) that comprise SONET networks were also discussed. Next, we defined network survivability in precise terms and described several survivable network architectures, with particular emphasis on survivable rings. This material serves primarily as background information for the work presented in subsequent chapters.

4. Capacity Analysis of Survivable Ring Architectures

4.1 Introduction

This chapter considers capacity-related aspects of survivable ring architectures. In particular, it explores two common subproblems in the design of ring-based transport networks: the Ring Sizing Problem and the Ring Loading Problem. The *Ring Sizing Problem* arises in situations where it is necessary to find the minimum capacity ring required to serve a stipulated set of demands. This may occur, for example, when a single ring is needed to connect a set of nodes in a metropolitan area network. The *Ring Loading Problem*, one the other hand, involves finding the subset of demands to load onto a ring of fixed capacity such that the total benefit (e.g., revenue) is maximized. This problem is motivated by a particular class of ring-network design algorithm, such as the one described in Chapter 8 of this thesis.

The chapter begins by discussing several factors that lead to different variations of these two problems. This is followed by a survey of prior work on the Ring Sizing Problem. The Ring Loading Problem, which has not been addressed in the literature, is then discussed and Integer Programming (IP) formulations are presented for four variants of the problem. Each variant represents different technology and policy choices for the deployment and use of BLSRs. The remainder of the chapter compares the ring loading efficiency of BLSRs under these choices in a series of statistical trials. The results of these trials are presented in Section 4.4, followed by some concluding remarks in Section 4.5.

4.2 Background

In practice, there are several variants of the Ring Sizing and Ring Loading Problems. These depend on the assumptions one makes about the demand model, loading policy and the ring technology attributes. The two most common demand models used in the literature are a static and dynamic demand model. In a *static* (or *off-line*) demand model, it is assumed that the entire set of origin-destination demands are known (or forecast) in advance and the (sizing or loading) problem is solved for all demands at once. A *dynamic* (or *on-line*) demand model is one in which the complete set of demands is not known in advance and routing and loading decisions must be made one at a time as new demands arrive (and possibly depart), without rearranging existing demands. While dynamic demand models are useful for evaluating on-line performance (e.g., probability of blocking), they involve additional complexities that are difficult to cope with in large-scale network optimization problems. Therefore, the majority of work surveyed here assumes a static demand model, as is usually the case in network design problems.

It is also a matter of operational policy as to whether demand bundles may be *split* or partitioned for provisioning. A *demand bundle* is the complete set of unit-demand bearing carrier signals (e.g., STS-1s or wavelengths) to be provisioned between a pair of nodes. There are two senses in which one may mean that demand bundles are split. One is in terms of the fraction of the complete demand bundle that may be selected for loading into a ring. This is sometimes required in SONET networks when the an entire group of carrier signals (e.g., DS-3s) need to have the same propagation delay from origin to destination, for example in inverse multiplexing applications. Unless otherwise noted, it is assumed here that each demand bundle is either loaded in its entirety into a given ring or not at all. The other sense of "splitting" has to do with whether the demand bundle may be split directionally around a BLSR. While directional splitting maximizes the loading flexibility for BLSRs, it is sometimes not preferred in practice for administrative, maintenance, and provisioning reasons. For example, some provisioning systems, such as the Bellcore ("Tirks") system [WuL90], do not allow demand bundles for the same origin-destination pair to be split around a ring. That is, all of the demand between a pair of nodes must be routed in only one direction within any ring. One of the objectives of the study in Section 4.4 is to assess the possible penalty of this simplifying operational policy in terms of its impact on the theoretically achievable ring loading efficiency.

Another issue in sizing and loading BLSRs is whether the ADMs are equipped for channel interchange. In a SONET ring, channel interchange (or *time-slot interchange*) allows the payload in one STS-1 time-slot on the incoming fiber to be mapped to a different STS-1 time-slot on the outgoing fiber. The equivalent functionality in WDM rings is *wavelength conversion*. In rings without channel interchange, each demand occupies the same channel (time-slot or wavelength) all the way around the ring from origin to destination. This mode of operation is known as *channel assignment*. With channel assignment, any two demands that traverse the same span cannot be assigned to the same channel. This has been called the *colour clash* constraint in the WDM context [EBC98]. When channel assignment is used, it is not always possible to utilize the full capacity of a BLSR, due to fixed channel assignment conflicts. To illustrate this, consider the channel assignment example in Figure 4.1.

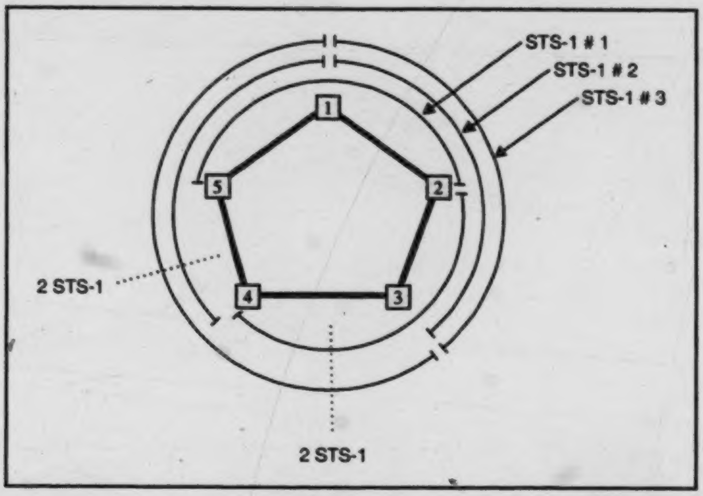


Figure 4.1. An example of channel assignment in a BLSR.

In this example, a SONET BLSR has one STS-1 of slack (or unused) capacity on span 3-4 in time-slot #2 and another on span 4-5 in time-slot #1. If the ADM at node 4 is not equipped for channel interchange, however, one STS-1 of demand between nodes 3 and 5 cannot be accommodated onto the ring because the slack capacity does not fall within the same time slot from origin to destination. In fact, there is no feasible routing and channel assignment plan for this example that allows the entire demand to be served.

Channel interchange eliminates this restriction, of course, but with added cost and complexity. Specifically, channel interchange complicates the APS protocol in a BLSR because channel mapping information has to be transmitted to intermediate nodes in the event of a failure. For this reason, the SONET standard for BLSRs [Bel95b] specifies that ADMs use channel (time-slot) assignment rather than channel (time-slot) interchange.

4.3 Ring Sizing Problem

Previous work on capacity-related aspects of survivable ring architectures has focused almost exclusively on the Ring Sizing Problem, which involves finding a set of routes for all demands that minimizes the required capacity of a ring. We prefer to use the term *sizing* here to describe this minimum size determination problem, although some literature refers to it as a *loading* problem. For UPSRs, the problem is trivial because there are no routing (or channel assignment) decisions to be made — each (bidirectional) demand circumnavigates the entire ring and, therefore, the required capacity is simply the sum of all demands on the ring. For BLSRs, the problem is more difficult because the required capacity depends on the direction in which each unit demand is routed around the ring, as explained in Section 3.7.2. The underlying demand pattern also has a bearing on the capacity requirements because it determines the degree to which ring capacity can be reused. The Ring Sizing Problem under idealized and general demand patterns is now discussed in the following sections.

4.3.1 Capacity Requirements for Idealized Demand Patterns

The capacity requirements of UPSR and BLSR rings under idealized demand patterns was studied by Cherng [Che91] for the case where demands may be split between the two directions around the ring. Closed-form expressions were derived for the four idealized demand patterns shown in Figure 4.2, which are representative of typical network applications.

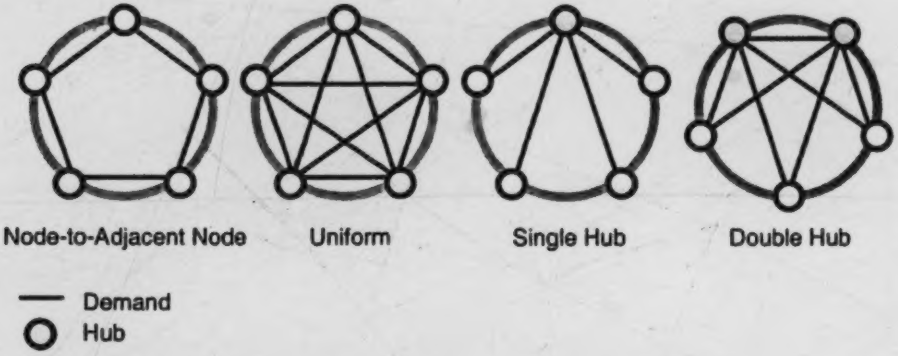


Figure 4.2. Idealized demand patterns (adapted from [Che91]).

In the *node-to-adjacent node demand pattern*, demands exist only between adjacent nodes around the ring. This demand pattern is typical of express rings that interconnect major cities in an inter-exchange subnetwork. The total number of demand pairs is equal to the number of nodes n in the ring. Therefore, the required capacity of a UPSR is

$$C_u = n \cdot d \tag{4.1}$$

where d is the amount of demand between node pairs. In a BLSR, the entire line section capacity can be completely reused from span-to-span, provided all demands are routed directly between adjacent nodes. In this case, the required (working + protection) capacity is

$$C_b = 2 \cdot d \tag{4.2}$$

A *uniform demand pattern* is one in which demand is uniformly distributed between all node pairs. This is common in core metropolitan and some long-haul subnetworks. The total number of demand pairs in a uniform demand pattern is $\binom{n}{2}$ and hence

$$C_u = \{(n^2 - n)/2\} \cdot d \tag{4.3}$$

In a BLSR, the minimum capacity occurs when demands are routed over the shortest path between each node pair. When the number of nodes in the ring is even, this involves splitting the demand between opposite node pairs and routing it in both directions around the ring. The required capacity of a BLSR with an even and odd number of nodes is

$$C_b = (n^2/4) \cdot d, \qquad n \text{ is even} \tag{4.4}$$

$$C_b = (n^2/4) \cdot d + d/2, \qquad n \text{ is odd} \tag{4.5}$$

Compared with the node-to-adjacent node demand pattern, the capacity requirements for a BLSR are greater because the average number of hops per demand increases and proportionally more capacity is required per demand. Or in other words, the opportunity for reusing capacity is diminished.

In the *hub demand pattern*, all demands are routed to a single hub node. Single hub demand patterns are typically found in access subnetworks. In this situation, the capacity requirements of a BLSR are the same as a UPSR because the working load on both spans incident to the hub node is equal to half the total demand. Here the total number of demand pairs is $(n-1)$ and the required capacity for both UPSRs and BLSRs is

$$C_u = C_b = (n-1) \cdot d \tag{4.6}$$

Because the demand in a single hub arrangement is susceptible to hub failure, two hubs are sometimes used for added survivability. In the *double hub demand pattern*, all demands are routed to two hub nodes and there is also demand between the two hubs. In the event of a hub failure, demand from the failed hub may be routed to the surviving hub. Here, the working load on the spans incident the hub nodes is equal to half the demand and therefore the capacity requirements of BLSRs and UPSRs are the same. Because the total number of demand pairs is $(n-2)$, the required capacity for both UPSRs and BLSRs is

$$C_u = C_b = (n-2) \cdot d \tag{4.7}$$

Figure 4.3 shows the capacity advantage of a BLSR relative to a UPSR for the four idealized demand patterns. The capacity advantage is the ratio of the demand carrying capacity of the BLSR to that of the UPSR.

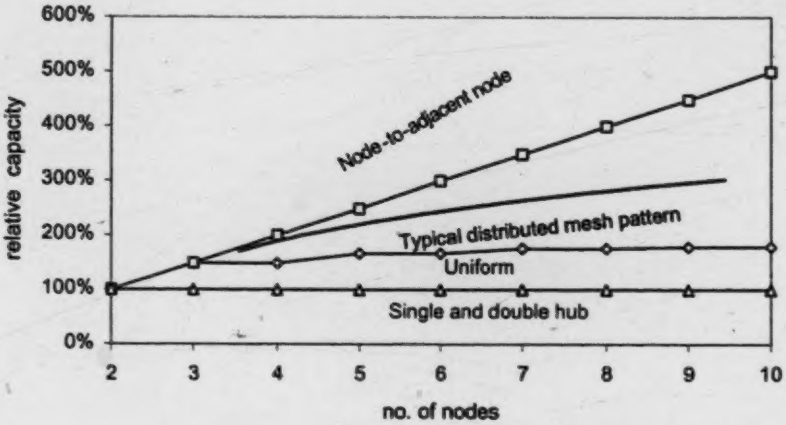


Figure 4.3. Relative BLSR demand carrying capacity for idealized demand patterns (adapted from [Nor96]).

In practice, most demand patterns usually lie somewhere between these idealized demand patterns. For typical distributed mesh patterns, for example, a two-fibre BLSR usually provides up to 300% more demand carrying capacity relative to a UPSR with the identical line rate [Nor96]. However, cost must also be considered when comparing BLSRs and UPSRs for a particular network application. Generally speaking, UPSRs have lower equipment costs than BLSRs due to their simpler protection switching mechanism. For this reason, UPSRs are usually preferred in access subnetworks where demands are typically hubbed and BLSRs are preferred in core metropolitan and long-haul subnetworks where demands are more evenly distributed [OwW93].

4.3.2 Capacity Requirements for General Demand Patterns

The Ring Sizing Problem for BLSRs under general demand patterns has been extensively studied. In these studies, it is generally assumed that the ring has full channel interchange at all nodes. For the case where demand bundles cannot be split between the two directions around the ring, the IP formulation of the Ring Sizing Problem can be expressed as follows:

Minimize: Z (4.8)

Subject to:

$$\sum_{k \in K_1(l)} d_k \cdot X_k^+ + \sum_{k \in K_2(l)} d_k \cdot X_k^- \le Z, \quad l = 1, \ldots, n \qquad (4.9)$$

$$X_k^+ + X_k^- = 1, \qquad \forall k \in D \tag{4.10}$$

$$X_k^+, X_k^- \in \{0,1\}, \qquad \forall k \in D \tag{4.11}$$

$$0 \le Z, \qquad \text{integer} \tag{4.12}$$

where Z is an integer decision variable for the capacity (size) of the ring, $D = \{d_k\}$ is the set of demands and d_k is the total quantity of demand (i.e., the size of the demand bundle) between nodes i_k and j_k and $i_k < j_k$. Note that the nodes in the ring are indexed from 1 to n around the ring and span i connects node i to node $(i+1) mod\ n$. X_k^+ is a binary decision variable that equals 1 if demand bundle k is routed in the clockwise direction and 0 otherwise, X_k^- is a binary decision variable that equals 1 if demand bundle k is routed in the counter-clockwise direction and 0 otherwise. $K_1(l) = \{k | i_k \le l < j_k\}$ is the subset of demands that traverse span l when routed in the clockwise direction and $K_2(l) = \{k |\ i_k > l \text{ or } j_k \le l\}$ is the subset of demands that traverse span l when routed in the counter-clockwise direction.

The objective function (4.8) is to minimize the line capacity Z of the ring. Constraint set (4.9) ensures that the sum of demands routed over each span (in both directions) does not exceed the line capacity. Constraint set (4.10) ensures that each demand bundle k is routed in either the clockwise or counter-clockwise direction. Constraint sets (4.11) and (4.12) assert that the routing decision variables are binary and the line capacity is an integer. Note that full channel interchange is implicit in this formulation simply by virtue of there being no constraints on the channels on which the individual demands may be routed around the ring. In other words if the capacity is adequate, a suitable channel interchange can always be found that accesses the required capacity in each span for each demand.

Cosares and Saniee [CoS94] showed that this IP problem is *NP*-hard by reduction from the partition problem [GaJ79]. They also developed several heuristic algorithms, including a weight-based, dual-ascent solution approach that is bounded above by twice the optimal solution [HJN96]. Since then several other heuristic algorithms have been proposed. For example, Schrijver et al. [SSW98] developed an efficient algorithm that exceeds the optimum by at most $1.5d_{max}$, where d_{max} is the largest demand. Building on this work, Khanna [Kha97] showed that, for any ε, a polynomial time algorithm exists that computes a solution within $(1 + \varepsilon)$ times the optimum.

For the case where directional splitting is allowed, Vachani et al. [VSK96] developed an efficient algorithm to compute optimal solutions in $O(n^3)$ time, where n is the number of nodes in the ring. Other efficient algorithms have been proposed by Lee and Chang [LeC97] and Myung et al. [MKT97]. The algorithm proposed by Myung et al. has a reported time complexity of $O(n|K|)$, where K is the set of demands.

Other work on the Ring Sizing Problem has considered the case where the rings have limited or non-existent channel interchange capability and the routing of demands is prescribed in advance (typically by shortest-path routing). In these studies, the problem is to find a fixed channel assignment that minimizes ring size, while satisfying the channel uniqueness (i.e., colour clash) constraint. The pure channel assignment problem (i.e., with no channel interchange) is equivalent to the problem of colouring circular arcs, which Garey et al. [GJM80] have shown to be *NP*-complete. Other work on this classical colouring problem by Tucker [Tuc75], shows that the upper bound on ring capacity (colours) is $(2w_{max} - 1)$, where w_{max} is the maximum number of unit demands (arcs) crossing any span. Note that with full channel interchange, the ring size is always w_{max}. An efficient algorithm for finding the optimal channel assignment for a uniform full mesh demand pattern (i.e., where each node requires a single wavelength to every other node) was proposed by Ellinas et al. [EBC98]. The channel assignment problem for rings (and other networks) with limited channel interchange capability is also considered by Ramaswami and Sasaki [RaS97], again for the case where demand routing is given in advance. They prove that a ring with channel interchange at only one node requires the same number of channels as a ring with full channel interchange. Other ring architectures with even more limited channel interchange are proposed that require at most $w_{max} + 1$ channels, i.e. only one more channel than would be required with full channel interchange.

Other variants of routing and channel assignment problems for optical networks are studied in [GSL98] and [RaS95]. An algorithm for dynamic channel assignment has also been proposed by Gerstel and Kutten in [GeK97]. A worst-case analysis of this algorithm shows that a ring without channel interchange may require as much as six times the capacity as one with full channel interchange for the same non-blocking performance. However, that result is for the case where shortest-demand routing is stipulated in advance for all demands.

4.4 The Ring Loading Problem

The nature of the Ring Loading Problem is to find a subset of demands to load onto a ring to maximize the total benefit (e.g., revenue) from the ring. For UPSRs, the problem is trivial and may be solved by loading demands in descending order of benefit per unit capacity until the ring is full. If demands cannot be split in loading the ring, the problem is equivalent to a 0-1 Knapsack Problem, which is known to be *NP*-hard [GaJ79]. For BLSRs, the problem is even more difficult because the demand carrying capacity depends not only on the ring's channel capacity but also on the routing and, where applicable, the channel assignment for all demands loaded into the ring. Thus, the problem not only involves the *selection* of demands to be loaded but also their respective routing and channel assignment choices within a ring that has a finite capacity.

To more fully appreciate the distinct orientation and motivation of this work from prior studies on the Ring Sizing Problem, it is helpful to consider that the loading problem, as defined above, arises in a particular class of approaches to the multi-ring network design problem. This approach, detailed more fully in Chapter 8, involves the iterative trial loading of a large number of ring candidate systems in each step of adding one ring to a growing multi-ring network design. Finding the best candidate ring (size, type, location, etc.) is crucially dependant on how well the ring is loaded from the *pool* of remaining unserved demands before it is assessed as a candidate ring in competition with all other candidates in that iteration. It is this design orientation that largely defines the present problem and gives it its relevance. The joint problem of specifying both the routing and channel assignment is also considered for this problem, where applicable. Unlike the Ring Sizing Problem, the Ring Loading problem has not been addressed in the literature.

One aim of this study is to assess the trade-off between technology cost versus loading efficiency in rings with and without channel interchange. Specifically, the question is; what loading penalty is incurred with channel assignment, relative to channel interchange, when optimal loading techniques are employed? The point is to quantify the theoretical benefit in loading efficiency that may be forfeited by the use of channel assignment only. This is a particularly relevant, and timely, question for WDM rings, for which standards are still under development. Specifying wavelength conversion could be an expensive requirement that may be of marginal practical benefit if optimal loading policies are adopted instead.

The second goal of the work is to quantify the effect of demand bundle splitting and channel interchange as a *policy* issue on loading efficiency, rather than to develop new algorithms for ring routing and/or channel assignment. To the best of our knowledge, this is the first systematic study of these aspects of the Ring Loading Problem, as defined herein and as distinct from the Ring Sizing Problem.

4.4.1 Loading Problem Formulations

This section presents IP formulations for optimal ring loading under the four ring loading disciplines. Formulations for two of the cases follow with only slight differences from their nearest corresponding cases so, for conciseness, the latter two of the detailed formulations are placed in Appendix A. These four loading disciplines or scenarios represent all combinations of loading policy (i.e., split versus non-split) and channel interchange capability (i.e., full channel interchange versus channel assignment).

By definition, for a non-trivial loading problem it is axiomatic for these purposes that the total demand in D exceeds that which can be completely served by any one ring. Because the ring is bidirectional, demands may be routed in either direction around the ring provided that the total load on any span does not exceed the ring line capacity c. For simplicity, we refer to the directions as "clockwise" and "counter-clockwise." The IP formulation for the Ring Loading Problem in which the nodes of the ring have channel interchange capability and demand bundles are not permitted to be split between the two directions around the ring, is as follows:

IP1 - Channel Interchange without demand splitting

Maximize:

$$\sum_{k \in D} d_k \cdot (X_k^+ + X_k^-) \tag{4.13}$$

Subject to:

$$\sum_{k \in K_1(l)} d_k \cdot X_k^+ + \sum_{k \in K_2(l)} d_k \cdot X_k^- \le c, \; l = 1, \ldots, n \tag{4.14}$$

$$X_k^+ + X_k^- \le 1, \qquad \forall k \in D \tag{4.15}$$

$$X_k^+, X_k^- \in \{0, 1\}, \qquad \forall k \in D \tag{4.16}$$

where X_k^+ is a binary decision variable that equals 1 if demand bundle k is routed in the clockwise direction and 0 otherwise. Likewise, X_k^- is a binary decision variable that equals 1 if demand bundle k is routed in the counter-clockwise direction and 0 otherwise.

The objective function (4.13) is the sum of demands routed in either the clockwise $(X_k^+ = 1)$ or the counter-clockwise $(X_k^- = 1)$ directions, i.e., this is the total demand served by the ring. Constraint set (4.14) ensures that the sum of demands routed over each span (in both directions) does not exceed its line capacity. Constraint set (4.15) ensures that each demand bundle *k* is routed in either the clockwise or counter-clockwise direction or not at all. That is, demand bundles are not split between the two directions. Note the last implication: if $X_k^{+} = X_k^{-} = 0$, it means that demand bundle *k* has simply not been selected from the demand pool for involvement in the loading of this ring. (In a complete network design it is implicit, however, that as the demand pool is depleted, eventually every demand bundle is served by some ring on each segment of its end-to-end route). Lastly, (4.16) asserts that the routing decision variables are binary.

The formulation for the Ring Loading Problem with channel assignment and demand bundle splitting can be written as follows:

IP2 - Channel Assignment with demand splitting

Maximize:

$$\sum_{k\in D}\sum_{t=1}^{c}(f_{kt}^{+}+f_{kt}^{-}) \tag{4.17}$$

Subject to:

$$\sum_{k\in K_1(l)} f_{kt}^{+} + \sum_{k\in K_2(l)} f_{kt}^{-} \le 1, \qquad l = 1, \dots, n,\ t = 1, \dots, c \tag{4.18}$$

$$\sum_{t=1}^{c}(f_{kt}^{+}+f_{kt}^{-}) = d_k \cdot X_k, \qquad \forall k \in D \tag{4.19}$$

$$f_{kt}^{-}, f_{kt}^{+}, X_k \in \{0,1\}, \qquad \forall k \in D, t = 1, \dots, c \tag{4.20}$$

where $f_{kt}^{+} = 1$ for a unit demand from bundle *k* that is routed over channel *t* in the clockwise direction and 0 otherwise. $f_{kt}^{-} = 1$ for a unit demand from bundle *k* that is routed over channel *t* in the counter-clockwise direction and 0 otherwise. X_k is a binary decision variable that indicates whether demand bundle *k* is selected for loading onto the given ring or not.

In this case, the objective function (4.17) is the sum, over all demand pairs, of the demands routed in both directions around the ring, recognizing that each channel number (time-slot or wavelength) exists in both the clockwise and counter-clockwise directions. This is the total demand served by the ring. Constraint set (4.18) specifies that each channel *t* can serve at most one demand on each span of the ring. By assigning separate f_{kt} variables for each combination of demand, channel, and direction, we are explicitly modelling the lack of a channel interchange functionality in this formulation. Note that the number of channels, indexed by t, is equal to the line capacity of the ring. Constraint set (4.19) ensures that for each demand bundle k, the sum of demands carried in both directions is either equal to its total demand d_k, $(X_k = 1)$, or zero, $(X_k = 0)$ (i.e., demand not selected for loading in this ring). As mentioned, there are actually four IP formulations which we consider for the ring loading problem. The other two IP formulations (i.e., channel assignment without splitting and channel interchange with splitting) are listed in Appendix A.

4.4.2 Study Method

By solving many trial instances of the previous formulations, we compared the performance of each of the four loading disciplines in terms of the total demand served for a variety of randomly generated trial cases. Each loading discipline was tested in four different ring configurations: (i) a 5 node ring with 12 channels, (ii) a 5 node ring with 48 channels, (iii) a 10 node ring with 12 channels, and (iv) a 10 node ring with 48 channels. For each ring configuration we generated 2000 random demand patterns, each of which served as the complete pool of available demands for one trial of all four IP problems. Half of the demand patterns were based on a random mesh demand model and the other half on a random hubbed demand model.

In the mesh demand model, the number of node pairs to have a non-zero demand was drawn from a binomial distribution with the probability of a non-zero demand being ½. For each non-zero demand bundle, the quantity of demand, d_k, was drawn from a discretized normal distribution with a mean of ½ the line capacity, c, and a standard deviation of ¼ c. In the event that a non-positive value was drawn from this distribution, it was discarded and another value drawn in its place. Likewise, any values greater than the line capacity of the ring were set to the line capacity for the respective loading trial. This represents that the ring under consideration can handle at most c of the larger prospective demand. The hubbed demand patterns were generated in a similar manner except that there were no demands directly between non-hub nodes. The quantity of demand between the hub node and all other nodes was also drawn from a normal distribution with the mean and standard deviation set to ½ c and ¼ c, respectively.

In most cases, these demand models generate a pool with enough total demand to permit a valid comparison of the competing loading disciplines. This is required for a meaningful study of the policy and technology issues at hand because one cannot assess the relative performance of loading disciplines when each of them is able to serve the *entire* pool of demand. Likewise, if the average size of demand bundles were arbitrarily large, then a subset of demands would frequently be found for each loading discipline that completely fills the ring, again obscuring any differences in loading efficiency that are truly due to the policy and technology choices involved. The model parameters were therefore chosen to produce random demand patterns in a range between these two extremes to enhance understanding of the effects attributable to loading policy. Within this range, the demand models produce a wide variety of demand patterns, in terms of the number of non-zero demand pairs and the bundle sizes. They also frequently generate demand patterns with a few very large demands, as in real networks. In summary, the statistical centering of the stochastic demand patterns reflects realistic patterns that we have seen in design studies, but is also legitimately set as a matter of experimental design so as to illuminate the intended phenomenon rather than try to study it in regimes where there really would be no policy or technology choices to make because rings were never fully loaded or always fully loaded.

For each ring configuration and random demand pattern we solved the optimal loading problems (IP1-IP4) using a parallel version of the CPLEX MIP Solver [CPL98] on a Sun UltraSparc HPC-450 with four processors, each running at 250 MHz. Typical run times were in the sub-second to second range, although some (primarily channel assignment) problems required a few minutes of run time.

In total, 8,000 random demand patterns (2,000 per ring configuration) were generated and the corresponding loading problems were solved for each of the four loading disciplines. As it turned out, 14% of the random demand patterns could be completely served by all four loading disciplines. To avoid an unintended source of bias, these results were excluded from the comparative results. In a further 1.2% of trial cases, an optimal solution could not be found for at least one of the loading disciplines within a five minute limit on execution time. These trials were also excluded from the results. To verify the accuracy of the results, sample solutions were drawn from the results and the feasibility of the routing and channel assignments were manually validated.

For each trial case, we used total demand served and loading efficiency as figures of merit to compare performance of each loading discipline in those cases (as mentioned above) where comparison is meaningful. *Loading efficiency* is defined as the total demand served normalized by the ring's "circumferential" capacity, i.e. the product of the ring's line capacity, *c*, and the number of spans in the ring, *n*. For example, a 10 node/48 channel ring has a circumferential capacity of 480 span-channels. Thus the relevant expressions for the loading efficiency for the split, η_s, and non-split $\eta_{n/s}$ cases with channel interchange are:

$$\eta_s = \left(\sum_{k \in D} (f_k^+ + f_k^-)\right) / n \cdot c, \tag{4.21}$$

$$\eta_{n/s} = \left(\sum_{k \in D} d_k \cdot (X_k^+ + X_k^-)\right) / n \cdot c, \tag{4.22}$$

This is a relevant measure of how effectively the ring resources (i.e., span-channels) are used to serve available demands. A loading efficiency of 1.0 would imply (and require) that all demands are served over a single span from origin to destination, as in the ideal "node-to-adjacent node" demand pattern. The higher the loading efficiency, the greater the quantity of (end-to-end) demand served per span-channel resource. Note that the loading efficiency, as defined, is not the same as the usual channel utilization (or fill). For example, a loading efficiency of ½ does not mean that 50% of the channels are being used. On the contrary, the corresponding channel utilization would generally be higher than 50% because demands typically traverse more than one span enroute from origin to destination.

The relative demand pool size was also used in assessing the performance of each loading discipline. Its relevance for characterization of these schemes is that it provides a measure of the pool size from which demands may be selected, relative to the ring's circumferential capacity. If a larger number of demands are available for loading a given ring then a higher loading efficiency should generally be attainable but this may vary for the different loading disciplines. Hence we record the relative demand pool size to inspect the results for this dependency. The *relative demand pool size* is the sum of all available demands in the demand pattern normalized by the ring's circumferential capacity, as follows:

$$|\hat{D}| \equiv \frac{1}{n \cdot c} \sum_{k \in D} d_k \tag{4.23}$$

To illustrate, a relative demand pool size of 1.0 means that the sum of all unit demands in the demand pattern is just equal to the circumferential capacity of the ring. To achieve a loading efficiency of 1.0 with this set of demands would require that all demands be adjacent node demands.

4.4.3 Results & Discussion

The most unexpected and significant finding in the results is that there is virtually no difference in the loading efficiency achieved with and without channel interchange. In the 3,431 trials based on the random hubbed demand model, there were actually *no* instances in which channel interchange permitted more demand to be served than did an optimal channel assignment solution working from the same demand pool. And in the remaining 3,372 trials involving the random mesh demand model, an increase in loading efficiency due to channel interchange was observed in only 15 trials. A histogram of the (log) frequency versus the percentage increase (or gain) in loading efficiency, $g = (1 - \eta_s/\eta_{n/s})$, for these cases is shown in Figure 4.4 for the random mesh demand model results. The histogram merges data for both the split and non-split trials. We return to discuss this interesting finding in Section 4.5.

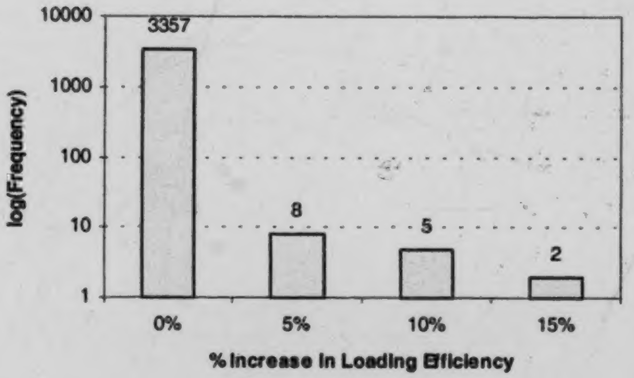


Figure 4.4. Gain in loading efficiency due to channel interchange (for random mesh demand pattern).

Next, to characterize the effect of demand bundle splitting we present the average gain in loading efficiency, $\bar{g}$, with and without splitting in Table 4.1. This table gives the average gain in loading efficiency due to splitting for each ring configuration and demand pattern model. Because the results for the trial cases with and without channel interchange were virtually identical, only the results for channel interchange cases are presented in detail. For the loading trials on the 5 node ring, Table 4.1 shows that demand splitting provided an average gain in loading efficiency of up to 14.8% relative to the non-split case. One-tenth of all test cases actually experienced gains in excess of 33% (the "90th percentile" results). In specific individual trials, the increase due to splitting was observed to be as high as 85%.

For the 10-node ring configurations, the gain due to splitting was less pronounced. The average gains in loading efficiency ranged between 1.3% and 2.8% for all cases. The 90th percentile of all test cases ranged from 4.3% to 7.3%. This reduction in gain with more nodes on the ring is attributable to two factors. First, the number of potential demand pairs increases as the square of the number of nodes in the ring. Consequently, there are simply more demand pairs from which to choose when loading the larger ring. This benefits the non-split loading more than the split loading discipline because the latter has a higher loading efficiency to begin with and thus has less margin for improvement. Second, the average-case difference in distance between the clockwise and counter-clockwise directions around the ring is more pronounced as the ring size increases. Consequently, proportionally fewer demand pairs can benefit from split routing because of the capacity penalty associated with routing demands the long way around the ring.

Table 4.1: Gain in loading efficiency by splitting (with channel interchange)†

		Mesh Demand Pattern		Hubbed Demand Pattern	
Nodes	Channels	$\bar{g}$	90th Percent.*	$\bar{g}$	90th Percent.*
5	12	11.7%	33.3%	13.7%	33.3%
5	48	13.6%	33.3%	14.8%	33.3%
10	12	1.9%	6.5%	1.3%	4.3%
10	48	2.8%	7.3%	2.7%	6.7%

* i.e., $(g^* | P(g \le g^*) = 0.90)$

† results without channel interchange are negligibly different.

For further insight into the results and to address possible issues of the experimental design vis-a-vis the centering of the stochastic demand patterns, scatter plots of the loading efficiency versus the demand pool size (normalized by ring capacity) are shown in Figure 4.5 and Figure 4.6 for the 5 node/48 channel and 10 node/48 channel ring configurations, respectively.

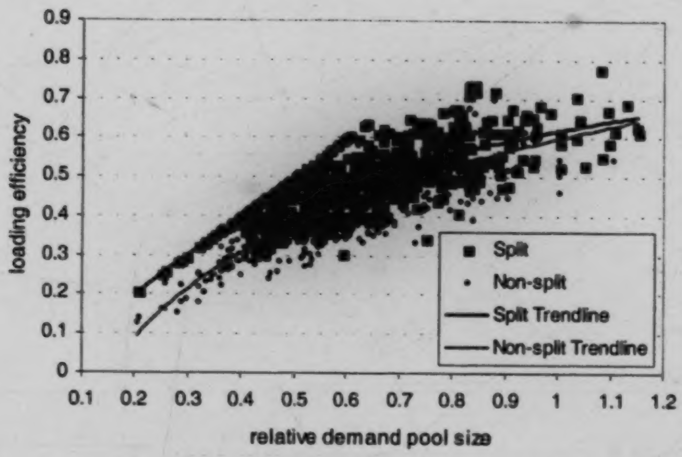


Figure 4.5. Scatter plot of loading efficiency vs. demand pool size for the 5 node/48 channel configuration, with a mesh demand pattern and channel interchange.

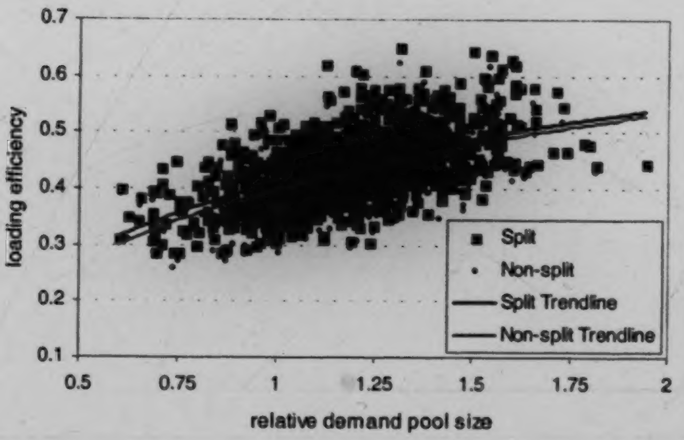


Figure 4.6. Scatter plot of loading efficiency vs. demand pool size for the 10 node/48 channel configuration, with a mesh demand pattern and channel interchange.

In these figures, the split and non-split loading efficiency for a given demand pattern is represented by a pair of data points on the vertical axis. As expected, the trend lines show that the average loading efficiency increases with larger demand pool sizes. However, the large variation about the mean indicates that the actual loading efficiency has significant detailed dependence on the specific demand pattern presented to the ring.

We also observe a significant drop in absolute loading efficiency as number of nodes on the ring increases, for the same relative size of demand pool to work from. For example, at a relative demand pool size of 0.8, the average loading efficiency (with splitting) for the 5 and 10 node rings is 0.56 and 0.36, respectively. This result is consistent with previous work [Fla90] that reports that larger rings are generally less efficient because, on average, demands must travel over a greater number of spans from origin to destination. They therefore consume proportionally more channel resources than those in a smaller ring per demand served between end nodes. In other words, in a large ring the propensity to have adjacent node demands is diminished relative to a smaller ring.

4.5 Summary

The potential benefits of demand splitting and channel interchange on ring loading efficiency are examined in this chapter. IP formulations were developed for four variants of the ring loading problem and used to assess the demand-serving implications associated with these policy and technology choices. Perhaps the most remarkable finding in the above results is that, when optimally planned, a ring with channel interchange provides a negligible advantage over one with no channel interchange in terms of the loading efficiency obtainable in a given demand environment. The complete absence of any cases where loading efficiency gains were seen in the trials involving a hubbed demand model also suggests that there may be some theoretical principle or generality underlying these results. Indeed, we show in Appendix B that when bundles may be split, a fixed channel assignment always exists that is equivalent to the best solution involving channel interchange.

Although channel interchange may allow proportionally more demands to be loaded in a dynamic (or on-line) loading environment, this has not been studied for the case where both routing and channel assignment are jointly determined. Nonetheless, it is a valuable insight for ring network design to know that channel interchange gives almost no advantage over an optimally planned channel assignment solution. This finding is particularly relevant to the ring network design method developed in Chapters 8. Specifically, it means that we don't need to solve the more complex channel assignment problem to evaluate the fitness of candidate rings because the channel interchange problem provides a suitably accurate estimate of loading efficiency for both cases. Actual channel assignment decisions, if required, can always be made off-line after the design is complete. This significantly reduces the time required to evaluate candidate rings and develop a complete multi-ring network design. This finding may also help telcos and standards organizations decide if the added cost and complexity of channel interchange is warranted. This may be especially relevant to WDM ring equipment development where wavelength conversion is a costly proposition.

On the other hand, the results show a significant benefit from a policy of permitting demand bundle splitting, especially in rings with relatively few nodes. Average-case gains in demand-serving capability up to 14.8% were observed for the five-node ring configurations, under the subset of test cases where the demand pool was not trivially satisfied by each ring and the demand pool was also not so high as to yield 100% loading efficiency in both cases. In other words, we believe this finding indicates generally significant benefits due to splitting in a range of realistic demand pools and demand patterns. Although the increase in demand served is dependent on the ring size and demand pattern, global increases in network efficiency may clearly be realized by permitting demand splitting.

5 The Multi-Ring Network Design Problem

5.1 Introduction

We now move from the problem of loading or sizing a single survivable ring to that of designing an entire network using survivable rings as the basic building block. We refer to this problem as the *multi-ring network design problem*. The chapter begins by defining the multi-ring network design problem and discussing its basic inputs, constraints and outputs. In Section 5.3, we consider the computational complexity of the basic problem and develop an approximate upper bound on the number of possible network designs. This is followed by a detailed discussion of several other important design considerations.

This chapter and the previous one together introduce the basic problems and issues in the multi-ring network design problem and set the stage for the main literature review that follows in Chapter 6. Some other literature on ancillary problems such as topology and dual-ring interconnection, however, is conveniently reviewed in this chapter. It is the most appropriate location to do so below because we do not return to these topics until much later.

5.2 Problem Description

The ring network design problem can be stated as follows: given a network graph $G = (N,S)$, a set of demands K, and a set of candidate ring technologies T, find a set of rings and a routing for all demands that minimizes the total design cost. A functional diagram of the overall multi-ring network design problem is shown in Fig. 5.1.

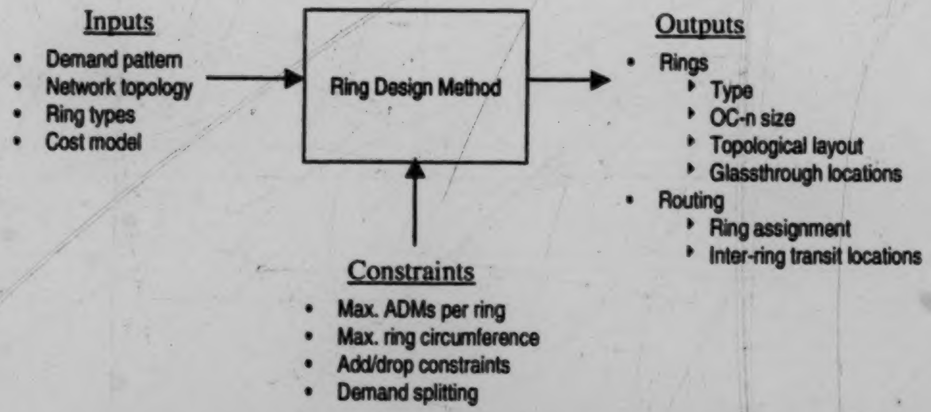

Inputs
Demand pattern
Network topology
Ring types
Cost model
Ring Design Method
Outputs
Rings
Type
OC-n size
Topological layout
Glassthrough locations
Routing
Ring assignment
Inter-ring transit locations
Constraints
Max. ADMs per ring
Max. ring circumference
Add/drop constraints
Demand splitting

Figure 5.1. Functional diagram of multi-ring network design problem.

In practice, there are several variants of this basic problem depending on the assumptions that one makes about the problem inputs, design objective and problem constraints. These factors are discussed in detail below.

5.2.1 Network Topology

The network graph G consists of a set of nodes N and a set of fibre spans S connecting the nodes. At a minimum, the network graph must be two-edge connected for a feasible ring design to exist. A *two-edge connected* graph has at least two edge (span) disjoint paths between every pair of nodes and, therefore, can potentially survive all single span failures. To protect against single node failures, the network graph must also be two-vertex connected. A *two-vertex connected* graph has at least two vertex (node) disjoint paths between every pair of vertices. Typically, the network graph is specified in advance based on existing or planned fibre spans (e.g., cable plant, duct structures or right-of-ways). Occasionally, however, the network planner may want to examine the impact of adding or deleting fibre spans from the network topology. Topology optimization is discussed in further detail in Section 5.4.3.

5.2.2 Demand Matrix

The set of demands or *demand matrix* specifies the amount of transmission capacity required between pairs of nodes in the network. These requirements may be measured in any one of a number of different demand units (e.g., DS1, DS3, STS-Nc). In some cases, there may also be constraints on whether the demand bundle may be split for routing purposes.Typically, the demand matrix is obtained by aggregating the demands from all client layers and converting them to the appropriate demand unit. Sometimes, this involves grooming and hubbing client layer demands to improve network utilization and realize economies of scale in the current network layer. Two types of grooming and hubbing in SONET networks are discussed in Section 5.4.2.

5.2.3 Ring Technologies

The candidate set of ring technologies specifies the types of rings that may be used to construct a design. Each candidate ring technology usually consists of a logical type (e.g., UPSR, BLSR) and a discrete modular line capacity (e.g., OC-12, OC-48). Clearly, the total flow on each ring cannot exceed its capacity, as described in Section 3.7. There may also be constraints on ring circumference, the maximum number of ADMs in the ring (e.g., 16) and the amount of capacity (or number of tributary signals) that can be added/dropped by each ADM. In some cases, only a single ring technology is considered in a network design. But in general it is usually advantageous to consider multiple ring technologies in the same design. These are called *multi-technology* designs. Considering multiple ring technologies, however, generally makes the problem more difficult to solve.

5.2.4 Design Objective

In practice, there are many attributes that determine the overall merit of a particular network design. Some of the more common attributes include total cost, ease of maintenance and administration, profitability, and growth potential. Thus, depending on the nature of the specific problem, the objective may be to minimize (or maximize) one or more of these (and other) design attributes. Problems that seek to optimize a collection of problem attributes are called *multi-objective optimization problems* [HeS82]. In most cases, however, a single design objective is chosen for simplicity.

In the multi-ring network design problem, the design objective is usually to minimize total design cost. The total design cost generally includes material costs such as fibre optic cable, regenerators, and ADM equipment and installation costs. These costs can be divided into fixed and variable costs [HoF91]. A *fixed cost* (or charge) is a cost that does not change with changes in a *cost driver* such as demand or distance. Fixed costs may include right-of-way costs, and ADM (and DCS) common equipment costs. Usually, fixed costs have a relevant range over which the relationship between the cost and the cost driver remains constant. For example, when the capacity of a ring is exceeded, additional fixed costs must be incurred. Thus, the fixed costs increase in steps as the demand (or distance) increases. A *variable cost* is a cost that changes in direct proportion to a cost driver. Variable costs may include fibre optic cable material and installation and ADM and DCS port costs.

An alternative approach is to use average costs to approximate the total design cost. *Average costs* are obtained by adding a portion of the fixed cost of the common equipment to the variable costs. Average costs are often referred to as unit costs, fully-allocated costs or pro-rated costs. The two typical cost drivers used for calculating average costs are demand and distance. Demand-related costs include ADM common equipment and port costs and, where applicable, DCS common equipment and port costs. Distance-related costs include fibre optic cable, signal regenerators, right-of-way costs, cable structures, and associated installation costs. In metropolitan networks, demand is generally a good predictor of total design cost because the cost of fibre and other distance-related costs are usually insignificant. In contrast, distance-related costs typically dominate the total design cost in long-haul networks. In intermediate or mixed networks, neither of these cost models alone is likely to accurately reflect the true cost of the network. An obvious alternative is to model cost as a linear function of both demand and distance. The main problem with using average costs, however, is that they ignore the step-wise increase in cost that occurs when the relevant range is exceeded (e.g., when the capacity of a ring is exceeded). As a result, the total design cost may be grossly underestimated when, for example, a ring serves relatively few demands.

It is also possible to approximate cost by using some other design metric that is closely correlated with cost. Capacity efficiency and demand capture are two such metrics. *Capacity efficiency* (or its inverse, *network redundancy*) is a measure of how efficiently ring capacity is being used within the network. Roughly speaking, the higher (lower) the overall capacity efficiency (redundancy), the lower the overall transmission related costs such as fibre and ADM-related costs. *Capture efficiency* is a measure of how well rings contain the demands that originate and terminate on them. The higher the capture efficiency, the greater the proportion of demand that is contained in the rings, and the lower the cost of managing inter-ring demand (e.g., DCS common equipment and ports). The effectiveness of any surrogate for cost depends on the degree of correlation between that surrogate and overall network cost.

5.2.5 Decision Variables

For each ring in the final design, the network planner must usually specify its logical type (e.g., BLSR, UPSR), capacity, topological layout and the locations of ADMs and glassthroughs. In addition, the routing pattern must specify the assignment of demands to rings, the direction in which each demand is routed within each (BLSR) ring, and the location at which demands transit from one ring to another when more than one ring is traversed from origin to destination. In general, however, the number and type of decision variables (or outputs) depends on the assumptions about the problem inputs and design objectives discussed above.

5.3 An Assessment of Problem Complexity

Clearly, the basic ring network design problem is a very difficult combinatorial optimization problem because solving it involves simultaneously finding the optimal ring set and routing pattern. This is particularly difficult because the two are highly interdependent on one another. That is, the ring set determines the feasible set of paths over which demands can be carried, while the routing pattern determines the spans (and their associated working capacity) that must be covered by the ring set. For this reason, the problem is usually decomposed into several subproblems. But in most (if not all) cases, these subproblems are *NP*-hard. For example, even if the optimal set of topological rings is specified in advance, the problem of finding the optimal routing pattern corresponds to solving an instance of a multicommodity capacitated network design problem, which is known to be *NP*-hard [GCF99]. Furthermore, the problem of finding the best cycle to connect the ADMs within each ring is equivalent to solving an instance of the TSP, which is also *NP*-hard.

Further insight into the combinatorial nature of the ring network design problem can be obtained by deriving an upper bound on the number of possible network designs. To do this we make the following simplifying assumptions. First, we assume that every node in the network originates/terminates at least one demand. We define a *network design* as a set of rings only, and ignore, for the moment, the pattern used to route the demand over the rings. We also assume that the design consists of only one type of ring (e.g., BLSR) and the capacity of each ring is unlimited relative to any aggregation demands routed over it (i.e., uncapacitated). Under these simplifying assumptions, a feasible network design is a set of rings which together cover every node in the network at least once.

The starting point for establishing an upper bound on the number of possible designs is to determine the number of candidate rings in the network graph. Here we define a candidate rings as a subset of active nodes (i.e., nodes equipped with an ADM) to be connected to form a ring. Therefore, the number of candidate rings is bounded above by the number of combinations of two or more active nodes within the network graph and is given by

$$Q \leq 2^{|N|} - |N| - 1 \tag{5.1}$$

where $|N|$ is the number of nodes in the network. For any combination of active nodes to correspond to a feasible ring, there must be at least one cycle within the network graph that connects the nodes. If the network has a Hamiltonian cycle, then every combination of active nodes is a feasible ring. This is because a Hamiltonian cycle visits every node in the network and, therefore, can connect any subset of active nodes within the network. Otherwise, the number of ring candidates will be less than the upper bound.

Next, to determine an upper bound on the number of possible network designs, we need a limit on the number of ring candidates in a network design. We note that when the rings are uncapacitated, the maximum number of topological rings required to cover every node in the network is $|N| - 1$. To prove this, consider the limiting case where every ring contains only two active nodes. To obtain maximum coverage while still meeting the connectivity requirement (i.e., that a path exists between every pair of nodes for inter-ring demands), every ring must have at least one node in common and one node disjoint from the other rings in the ring cover. In this situation, the first ring covers two nodes and the remaining $|N| - 2$ rings cover one additional node, for a total of $|N| - 1$ rings. From a connectivity point of view, any additional rings placed on the network beyond this point are redundant because all nodes are already covered and the network is fully connected. Based on this argument, the upper bound on the number of possible network designs is

$$\sum_{i=1}^{|M|-1} \binom{Q}{i} \tag{5.2}$$

A graph of the number of possible designs as a function of the number of network nodes is shown in Fig. 5.2. This figure shows that the number of possible network designs is grows very quickly for networks with more than a handful of nodes. Consider, for example, a Hamiltonian network with ten nodes. From Eqs. (5.1) and (5.2), the number of candidate rings and the number of possible network designs are roughly 10^{13} and 10^{21}, respectively. Even if we could evaluate a million designs per second, it would take nearly 100 million years to evaluate every possible network design. Admittedly, this includes the evaluation of network designs that are not feasible. But it is not clear how to find the globally optimal solution without evaluating all network designs because the problem is *NP*-hard, so this bound is still quite relevant.

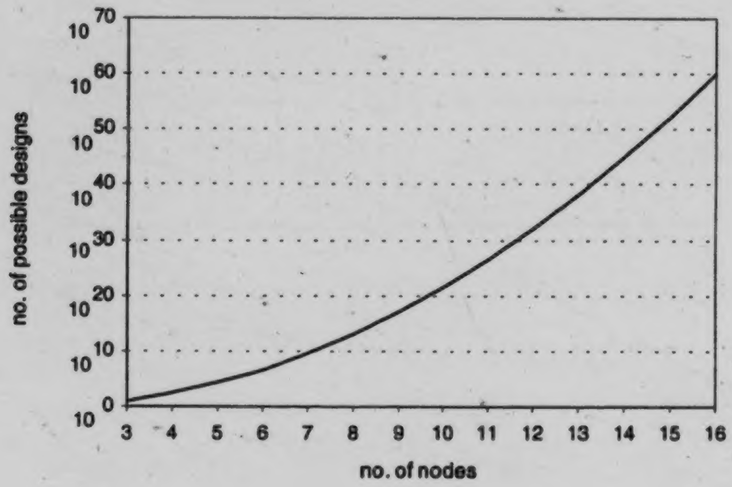


Figure 5.2. Upper bound on the number of possible designs.

5.4 Other Design Considerations

In the preceding sections we considered a basic definition of the ring network design problem without discussing some of the underlying assumptions and other pertinent design considerations. These other considerations are discussed in detail below and will later serve as a framework for evaluating the design methods discussed in Chapters 8 through 10.

5.4.1 Decision Environment

An important aspect of any network planning exercise is understanding the environment in which decisions are being made. A key characteristic of this environment is whether the problem is being evaluated for a single period or multiple periods. In a *single-period* or *static* decision environment, all decisions are made for the current period only, independent of any future periods. The definition of the ring network design problem in Section 5.2, implicitly assumes a static decision environment.

In many practical network planning applications, however, decisions about where and when to deploy network capacity may be considered over multiple time periods. In these situations, the goal is to find the optimal sequence of decisions over the planning horizon (i.e., the duration of the problem). This is referred to as a *multi-period* or *dynamic* decision environment. In the context of the ring network design problem, for example, the deployment of rings in one period usually has an impact on the timing, capacity and cost of any rings deployed in future periods. Quite often, the ring network design problem may also involve augmenting the capacity of an existing network (sometimes within a fixed capital budget) to accommodate growing demand. This is called *transition planning*. Although a multi-period model may more accurately reflect the actual decision environment in which network planners operate, the introduction of temporal dependence usually complicates a model's optimization. That is, the optimization of an *n*-period model usually requires even more time and computer storage than *n* separate single-period problems.

In some cases, a multi-period problem can be transformed so that the consequences of present decisions can be safely ignored in future periods. Under these circumstances, the optimal solution is obtained by solving a series of static problems, one for each time period. Problems that possess this property are called *myopic optimum problems*. Whether the multi-period ring network design can be transformed to a myopic optimum problem is an open question.

Another characteristic of the decision environment is whether it is deterministic or probabilistic. A *deterministic decision environment* is one in which the uncertainty surrounding the outcome of any one decision is so small that it can be ignored. In the context of the multi-ring design problem, this means that important design information such as the demand pattern, equipment costs, and technological advances are known or can be forecast with little uncertainty and that any variations in these factors do not seriously affect the outcome. If this is not the case, then a probabilistic model may be more appropriate for comparing alternative decisions. A *probabilistic model* accounts for the probability of achieving specific outcomes for a given set of decision alternatives. But including uncertainty in a model makes the model more complex. Consequently, most (if not all) methods for the ring network design problem assume a deterministic decision environment. As a practical alternative, sensitivity analysis may be used to evaluate the effect of forecast errors on prospective designs.

5.4.2 Demand Grooming and Hubbing

When the volume of demand between two nodes is sufficient to fill or almost fill an entire STS-1 time slot, a *direct STS-1* is usually assigned to the node pair. In a UPSR, this means that the STS-1 time slot is not accessed by any other nodes on the ring. In a BLSR, a direct STS-1 time slot cannot be accessed at intermediate nodes enroute to the destination, but can be reused once the demand reaches its destination. When the demand between a pair of nodes is relatively small (e.g., only a few DS1s), however, a direct STS-1 is not very efficient because most of its capacity is unused.

An alternative is to consolidate or *groom* the demand to achieve better utilization. Grooming involves rearranging and repacking client layer demands (e.g., DS1s) into STS-1 time slots to obtain high utilization or fill ratios. Grooming allows demand from various destinations to be combined together over a single transport facility. In ring-based networks there are two ways to groom demand: centralized demand management; and distributed demand management. In *centralized demand management*, also known as *hubbing*, all DS1 demand, for example, originating from a node is packed into one or more STS-1s and transported to a central hub node. At the hub node, the incoming STS-1s are dropped from the ring and connected to a digital cross-connect system (DCS). Within the DCS, the individual DS1s with the same destination are *groomed* into outgoing STS-1s, which are added back to another ring and transported to their common final destination.

One disadvantage of hubbing is the difficulty in accommodating growth because there is a propensity to continue hubbing demands even after they reach direct STS-1 levels. The reason for this is services must be interrupted to switch from hubbing to direct STS-1s. In addition, hubbing introduces a single point of failure (i.e., the hub node). This situation can be improved immensely by introducing a second hub node and splitting demand between the two hub nodes. Studies by Owen and Wulf-Mathies [OwW93] show that in hierarchical UPSR networks, dual hubbing requires very high capacity rings at the upper levels. Although less capacity is required if BLSRs are used at the upper levels, the general hubbed nature of the demand pattern limits the capacity advantage of BLSRs relative to UPSRs (as described in Chapter 4).

An alternative to hubbing is distributed demand management. In *distributed demand management*, DS1 demands are routed directly within the ring over a shared or *collector* STS-1. A collector

STS-1 is an STS-1 time slot that is accessible by more than one pair of nodes. In order to access a collector STS-1, however, an ADM must be capable of time slot assignment (TSA) at the VT1.5 level. Unlike for hubbing, distributed demand management does not have a single point of failure and also does not incur a capacity penalty due to back-hauling. ADMs with VT1.5 accessibility, however, are generally more costly than those without this feature.

A hybrid demand management scheme using direct STS-1 and grooming is also a possibility. In a *hybrid scheme*, demands above a certain DS1 threshold are routed over direct STS-1s, whereas demands below the threshold are groomed using either centralized or decentralized demand management techniques.

Clearly, the type of demand management scheme and the capabilities of the ADMs have a direct impact on ring capacity and ring loading. For example, if direct and collector STS-1s are used in a BLSR, the ring loading subproblem must be solved using multiple transport signal units (e.g., DS1 and STS-1 units) to find the optimal routing for each STS-1 and DS1 demand. Ring loading with multiple transport signal units is also applicable to the problem of finding the optimal routing of single and concatenated STS-1 payloads (e.g., STS-3c, STS-12c). If the total demand (i.e., all of the demand entries) between a pair of nodes cannot be split, these problems can be solved using single transport signal unit algorithms by expressing each demand in the smallest unit. For example, if a node pair has a total of two STS-1s and four DS1s of demand, the aggregate demand can be expressed as 60 DS1s. If each demand entry between a pair of nodes can be split, a practical alternative is to bundle these demands up to the largest unit and solve the problem for the single larger unit only. For the previous example, the aggregate demand would be expressed as 3 STS-1s. The bundling procedure, however, would in general degrade loading efficiency due to the larger granularity.

5.4.3 Topology Optimization

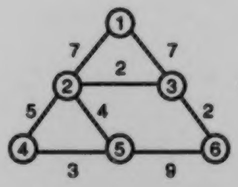
The complexity analysis in Section 5.3 assumes that the underlying network topology is already given. More generally, it may be desirable to identify the combination of new and existing spans that minimize the total network costs. This is referred to as *topology optimization*. The optimal topology for ring-based designs depends strongly on the demand pattern and the cost structure of the problem.

To illustrate this consider the demand pattern, network graph and two alternative network designs shown in Fig. 5.3. The aggregate demand on each span in Fig. 5.3(a) was obtained by routing each demand over the shortest path between the origin and destination nodes. Typically, route length is usually measured in terms of its hop count (i.e., the number of spans in the route), cumulative geographical distance, or the sum of some other weight (e.g., cost) associated with each of its spans. Routing demands over the shortest or least-cost route is a reasonable approach because it generally consumes the least amount of transmission capacity. There are cases, however, where shortest path routing leads to sub-optimal results if some ring systems are already in place or are committed to an evolving design. In Fig. 5.3(a), shortest path routing results in every span in the network graph being assigned some working demand. If we assume that 4B12 rings are being considered, a min-cost design requires two rings and nine Add/Drop Multiplexers (ADMs) to cover every span in the network graph, as shown in Fig. 5.3(b). Note that other min-cost span coverage designs exist with the same number of rings and ADMs.

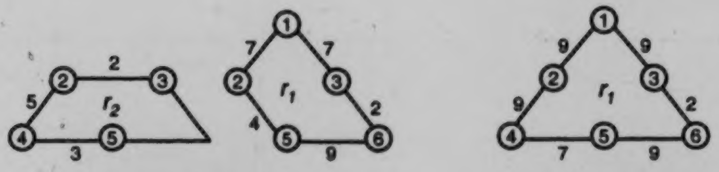
Demand Matrix (in STS-1s)

To

	Node	1	2	3	4	5	6
	1	X	3	6	0	4	1
	2		X	2	5	0	0
From	3			X	0	0	1
	4				X	1	2
	5					X	7



(a) Aggregate demands using shortest path routing.



(b) A ring design based on shortest path routing.

(c) A ring design based on alternate routing.

Figure 5.3. An example of demand routing in multi-ring networks.

If we reroute the demand on spans 2-3 and 2-5 (i.e., demands 1-5 and 2-3) around the periphery of the network, however, we can eliminate the need to cover spans 2-3 and 2-5. In this situation, a single ring with six ADMs is sufficient to carry all network demands, as shown in Fig. 5.3(c). Rerouting working demands to avoid having to cover a span is called *span elimination.*

Several approaches for span elimination were recently studied by Lee et al. [LGM99], [Lee00]. In total, four span elimination heuristics were developed and tested. Two of the heuristics attempt to optimize network topology (and the resulting multi-ring network design) using a variant of shortest path routing in which the effective distance (or cost) of each span is a function of its working load and the ring module size. Demands are routed one at-a-time and at each iteration the span costs are updated based on their working load. By discounting the cost of spans that are lightly loaded (relative to the ring module size), the algorithm tends to aggregate demands onto fewer spans. After all demands have been routed, the cost of the topology is evaluated by creating a multi-ring network design using the SCIP formulation described in Section 9.3 of this thesis. One heuristic, called *modular aggregated pre-routing*, uses a fixed cost function and, as a result, creates only one network topology. In the other heuristic, called *iterative modular aggregated pre-routing*, uses a family of cost functions to create several network topologies, the best of which is selected as the solution.

The other two heuristics begin by routing the demands over the shortest path and then evaluating several possible span elimination combinations. The first heuristic, called *iterated routing and elimination*, uses a breadth-first search procedure to successively eliminate spans from the initial topology. At each iteration, the heuristic evaluates all possible span elimination candidates by temporarily eliminating the span, rerouting the affected demands and determining the cost using the SCIP formulation. The span elimination candidate with the lowest cost is selected and permanently eliminated from the topology. This process continues until no further spans can be eliminated from the topology without violating the two-edge connectivity requirement, as discussed in Section 5.2.1. The second heuristic, called *post inspection and rerouting*, works in a similar manner except the SCIP formulation is not used at every iteration to evaluate span elimination candidates. Instead, it begins by creating a multi-ring network design for the initial topology using the SCIP formulation. It then attempts to reroute demands to minimize the number of supermodular spans. A *supermodular* span is one on which the working load exceeds the capacity of the largest modular ring placed on that span. At each iteration, the span elimination candidate that results in the fewest number of supermodular spans is eliminated from the topology. This process continues until no span can be eliminated without increasing the number of supermodular spans or violating the two-edge connected requirement.

The results showed that the iterated routing and elimination heuristic consistently generated the best solutions (with savings of up to 26% relative to the initial network topologies) but with much longer runtimes. Overall, the post inspection and rerouting heuristic provided the best tradeoff between solution quality (with savings up to 18%) and runtime.

5.4.4 Dual Ring Interconnect

In multi-ring networks, *intra-ring* demands are protected against single node and span failures by the protection switching mechanism of the ring. *Inter-ring* demands, however, are susceptible to failures at nodes where the demand transits from one ring to another. This may be acceptable for the majority of demand since the probability of transit node failures is relatively low. For certain classes of demand (e.g., military applications, common channel signalling), however, single points of failure may not be acceptable due to high availability requirements. To meet these requirements, a *matched node* arrangement has been adopted as a SONET standard [Bel95a], [Bel95b]. The matched node arrangement for BLSRs is shown below in Fig. 5.4.

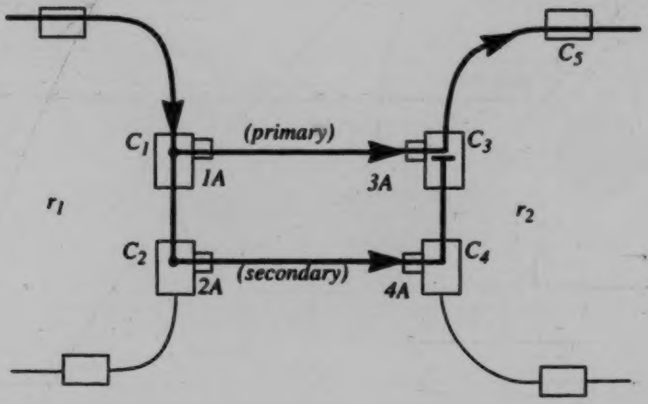


Figure 5.4. Matched node drop & continue inter-ring transfer arrangement for BLSRs (shown for one signal direction only).

In this arrangement, an inter-ring demand is protected by establishing redundant signal paths between the adjacent rings that serve the demand, as shown in Fig. 5.4. The redundant signal paths are made between a pair of primary and secondary inter-ring gateways (i.e., ADMs) in each ring. The *drop and continue* feature of SONET ADMs is used to drop a copy of the tributary signal (e.g., STS-1) at the primary gateway C_1 in ring r_1 as it passes through enroute to the secondary gateway C_2. At C_2, the tributary signal is dropped from the line signal and passed to its adjacent gateway C_4 in ring r_2. Here it is added to the line signal in ring r_2 and routed to C_3. At C_3, a *service selector* is used to switch from the primary to the secondary signal in the event a signal failure or degradation. In some ADMs, the drop and continue signal between the primary and secondary gateway may be carried on protection capacity. This conserves working capacity for demands. For simplicity, only one signal direction is shown in Fig. 5.4. The same arrangement protects the other signal direction.

Note that because the protection switching in a matched node arrangement is performed on a path (e.g., STS-1) basis, each inter-ring demand has its own pair of primary and secondary gateways. Thus, the primary gateway for one demand may be the secondary gateway for another demand and vice versa. In addition, the protection switching mechanism in one ring is completely independent from the other ring. As a result, a primary gateway in one ring may feed either a primary or a secondary gateway in the adjacent ring. Typically, these adjacent gateways are located within the same central office building. However, when two adjacent gateways are not co-located, linear systems and/or other intermediate rings may be used to carry the inter-ring demand. This is called an *extended matched node*. The primary and secondary gateways within the same ring may also be separated by one or more intermediate nodes. It is usually more capacity efficient, however, if the two nodes are adjacent [DDH97].

In a UPSR, the matched node arrangement operates in a slightly different manner, as shown in Fig. 5.5.

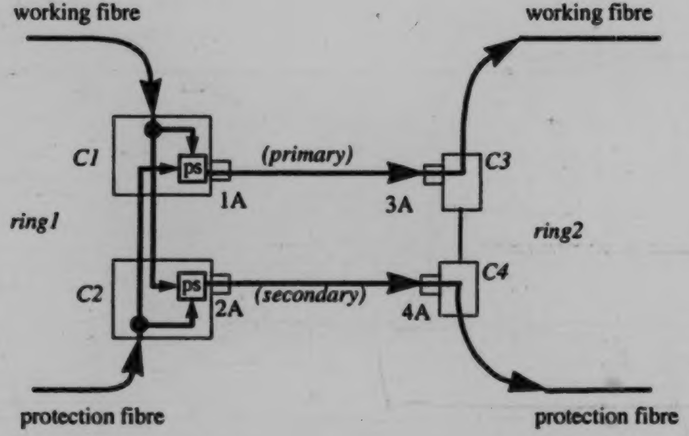


Figure 5.5. Matched node drop & continue inter-ring transfer arrangement for UPSRs (shown for one signal direction only).

Here the drop and continue feature is used at both the primary and secondary gateways. Because the working and protection signals are normally routed all the way around a UPSR, no additional capacity is required for the arrangement. At the primary gateway C_1 in ring r_1, the incoming signal on the *working* fibre is dropped as it passes through to the secondary gateway C_2. Likewise, the incoming signal on the *protection* fibre at C_2 is dropped and then continued to C_1. At both gateways, a path selector is used to select between the working and protection signals, as required. The selected signal is then routed to the adjacent gateway on ring r_2 where it is routed over the protection or working fibres to its final destination. For simplicity, only one signal direction is shown in

Fig. 5.5. The mirror image arrangement protects the other signal direction.

Like the BLSR matched node arrangement, a primary gateway in one ring may feed either a primary or a secondary gateway in the adjacent ring. In addition, hybrid BLSR/UPSR matched node arrangements are also possible because each ring in a matched node arrangement acts autonomously from its adjacent ring. When inter-ring demands travel over more than two rings, matched nodes are required between all intermediate rings along the path to provide end-to-end protection against single-point failures.

An alternative, recently considered, arrangement for protecting inter-ring demand from transit node failures is the dual-feeding arrangement [Gro97a] shown in Fig. 5.6.

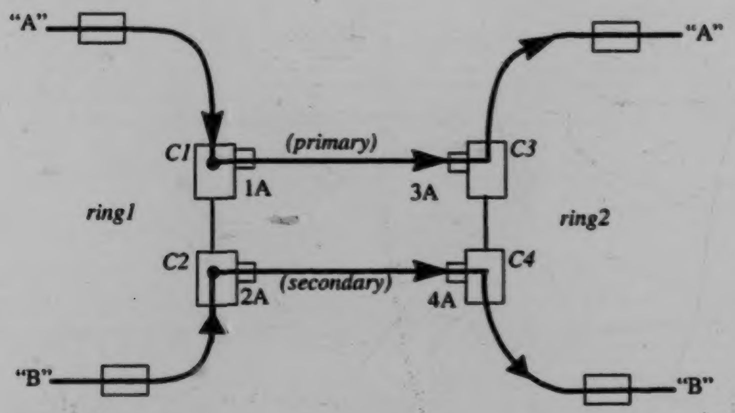
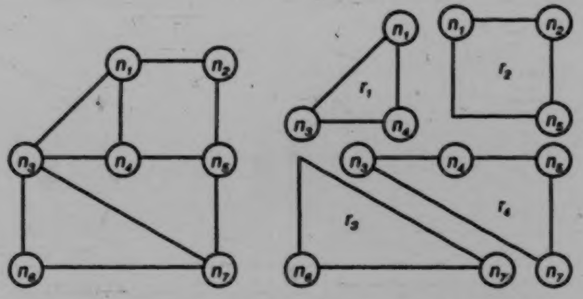


Figure 5.6. Dual feeding inter-ring transfer arrangement.

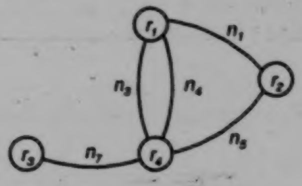
In the dual feeding arrangement, the tributary signal is simply duplicated and sent in both directions around the ring to two separate gateway nodes. Note that dual feeding is completely compatible with the matched node arrangement because inter-ring interface for both arrangements is the same (i.e., two separate signal feeds) and protection switching actions are made independently by each ring. In fact, any combination of matched node and dual feeding sections may be used to protect end-to-end demands against single-point failures.

Clearly, the dual feeding and matched node arrangements consume more capacity than an unprotected inter-ring transition. The trade-offs between dual feeding and matched nodes has been studied by Grover [Gro96], [Gro97a]. These studies show that the worst-case survivability of both approaches is comparable. In terms of capacity efficiency, the matched node arrangement is usually more efficient when distance between the primary and secondary gateways is small in comparison with the maximum distance from either node to the origin. Otherwise, the dual-feeding arrangement tends to be more capacity efficient. The problem of finding high-availability paths in ring-based networks using combinations of dual-feeding and matched node arrangements has also been studied by Grover [Gro99] and Sui [Sui99].

Both arrangements also introduce the added constraint that every ring requiring matched nodes must be connected to at least one other ring at a minimum of two nodes. Two rings are said to be *connected* at a node if they are both equipped with an ADM at that node. The connectivity requirement can be verified for a given ring cover R by representing it as an undirected *ring connectivity graph* G'. In a ring connectivity graph G', each ring r_i is represented by a vertex and each node (e.g., central office) at which two rings are connected is represented by an edge. Since two rings may be connected at more than one node, G' may have multiple parallel edges between vertices. To satisfy the ring connectivity constraint for all possible demand pairs, G' must be biconnected. That is, there must be at least two edge-disjoint paths between every pair of vertices in G'. An example of a ring connectivity graph is shown in Fig. 5.7.



(a) Network graph (b) Ring design



(c) Ring connectivity graph

Figure 5.7. An example ring connectivity graph.

An example of a network graph and corresponding ring design is shown in Figs. 5.7(a) and 5.7(b), respectively. The ring design consists of four rings: r_1, r_2, r_3, and r_4. Note that two of the rings, r_2 and r_3, contain glassthroughs at nodes n_4 and n_3, respectively. The connectivity between rings is represented by the ring connectivity graph shown in Fig. 5.7(c). In Fig. 5.7(c), each of the rings is represented by a vertex and the nodes at which they overlap are represented by edges between the vertices. For example, in Fig. 5.7(b) we note that ring r_1 is connected to ring r_2 via node n_1. This is represented in Fig. 5.7(c) by a single edge n_1 between vertices r_1, and r_2. Similarly, ring r_1 and ring r_4 are connected via nodes n_3 and n_4. This is represented in Fig. 5.7(c) by two parallel edges n_3 and n_4 between vertices r_1, and r_4. From the ring connectivity graph it is easy to see that the ring cover is not biconnected. That is, if node n_7 fails, ring r_3 will be isolated from the rest of the network.

5.5 Summary

This chapter provided a generic definition of the ring network design problem, assessed its computational complexity and discussed several other relevant factors. In the next chapter, the material covered in this chapter serves as a framework for reviewing the prior work on the ring network design problem.

6 Related Work

6.1 Introduction

This chapter surveys of prior work on the single and multi-ring network design problems. These problems have been studied extensively in recent years and a large and growing body of research already exists on this topic. This is due in part to the wide variety of ring network planning scenarios that exist in the real world and an abundance of solution techniques for addressing these types of problems. Because a comprehensive survey of all work in this area is beyond the scope of this thesis, we provide a representative sample of the major variants of the problem and solution techniques that have been studied to date. For a description of other work on this research topic see [MoG98b], [SWS99].

As shown in Figure 6.1, ring network design problems can be divided into two main classes: single ring network design problems and multi-ring network design problems. The single ring network design problem involves finding a single ring to connect a set (or subset) of nodes in a network. This type of problem occurs most frequently in the context of a metropolitan area network design. Generally, the objective of this problem is to maximize revenue (or profit) or to minimize total construction cost. Five methods for the single ring network design problem are covered in Section 6.2.

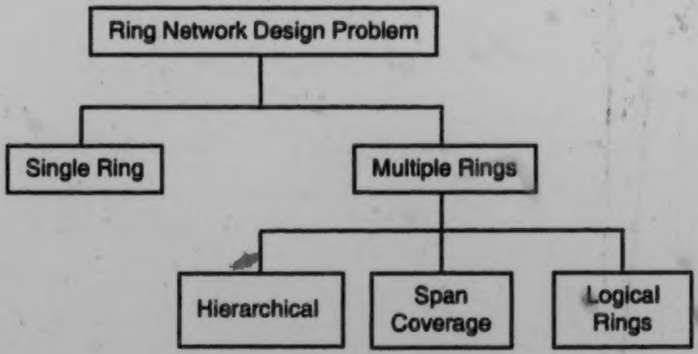


Figure 6.1. Classification of Ring Network Design Problems.

A second class of problems occurs when the network design consists of multiple rings overlying different topological cycles. These problems can be further subdivided based on a number of other attributes of the problem or decision environment including network equipment constraints, network application, company operating policy and other considerations (e.g., planning horizon, capacitated vs. uncapacitated, metro vs. long-haul, etc.) discussed in Chapter 5. Here we classify them according to the approach used to decompose the main problem into more tractable subproblems. As discussed in Chapter 5, some form of decomposition or relaxation of the main problem is usually necessary due to the complexity of the problem. One approach is to divide the design into two or more *hierarchical* layers and solve each layer independently. Two of the methods surveyed in Section 6.3 fall into this category.

Another popular approach is to first route the demands over the network topology and then select an optimal set of rings that *covers* (or carries) the working load assigned to each span in the network graph. Generally, the objective in this case is to minimize total design cost or network redundancy. We refer to this problem as the *span coverage problem.* A property of the span coverage approach is that the resultant design cost may depend heavily on the pre-determined demand routing pattern. There are two reasons for this dependence. The first is that rings are usually available in a discrete set of modular capacities only. As a result, small changes in the routing pattern can cause a step-wise increase in the design cost when ring system capacity is exceeded or when very lightly loaded spans arise, each of which may trigger the placement of a new ring to satisfy the coverage imperative. To illustrate, consider the example in Figure 6.2.

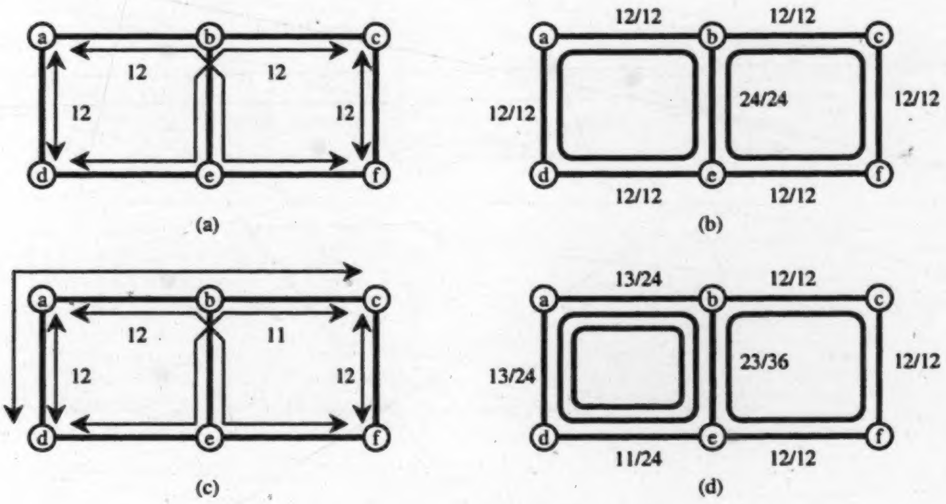
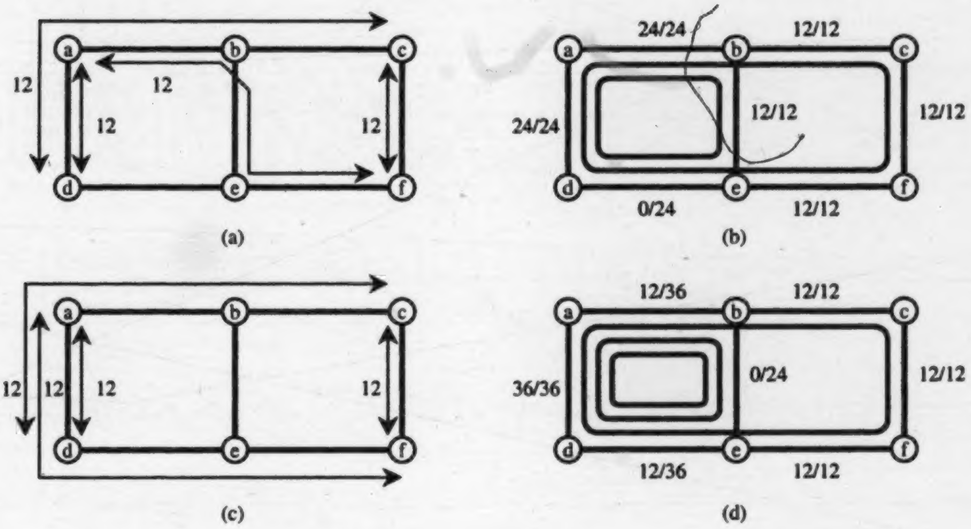


Figure 6.2. Effect of ring modularity on transmission capacity requirements.

Figures 6.2(a) and 6.2(c) show two possible routing patterns for a six node network with four demand pairs: a-d, a-f, c-d, and c-f. Each demand pair requires 12 DS3s between its end nodes. In Figure 6.2(a), the demands are routed over the shortest path (measured in hops) between the origin and destination nodes. If the ring capacity is also 12 DS3s, it is easy to see that only two rings are required to cover the routing pattern, as shown in Figure 6.2(b). The capacity utilization (working load + aggregate capacity) for each span is also shown in Figure 6.2(b). In this case, the aggregate capacity is completely utilized on all spans. The routing pattern in Figure 6.2(c) is identical to that shown in Figure 6.2(a) except one DS3 from demand pair c-d is rerouted around the perimeter of the network graph on spans a-d, a-b and b-c. Although the total length of this new route is the same as the original route, the total load on spans a-b and a-d now exceeds the modular capacity of the ring by one DS3. In this case, the minimum number of rings required to cover the routing pattern increases from two to three. As a result, the total installed capacity goes from 96 DS3-hops to 144 DS3-hops and the average utilization (or fill) drops from 100% to only 66.67%. This example demonstrates inefficiencies that can occur due to ring modularity effects alone.

Even when the working load on each span is a multiple of the ring's capacity, however, there still may not be an efficient way to cover the spans with rings. That is, the topology may not support a ring set that closely matches the span usage and capacity requirements of the routing pattern. To illustrate consider the routing patterns and ring covers shown in Figure 6.3.



(a) (b) (c) (d)

Figure 6.3. Effect of topological layout on transmission capacity requirements.

Here, the network graph and demand matrix are the same as in Figure 6.2 but the routing patterns are modified slightly. In Figure 6.3(a), all of the demand between demand pair c-d is routed around the perimeter of the network. Although the load on each span is an even multiple of the ring capacity, there is no set of rings whose aggregate capacity exactly matches the span loads. And although the spans can still be covered by two rings, the total installed transmission capacity increases from 96 DS3-hops to 120 DS3-hops. A similar situation is depicted in Figures 6.3(c) and 6.3(d) except the inefficiencies are even more pronounced. Note that in all examples shown above, demands are routed over the shortest path. Although these examples are somewhat contrived, they clearly demonstrate the effect that demand routing can have on a network's total capacity requirements and ultimately its total design cost. Four of the twelve methods surveyed in Section 6.3 use this approach.

A third approach is to establish *clusters* of nodes to be connected by a set of rings and then find the optimal fibre routing and demand assignment (and routing) for each cluster. We refer to these clusters as *logical rings*. Several heuristics have been proposed using this approach. In general, the logical rings may be specified in advance or determined by evaluating the *community of interest* between the nodes in the network. The community of interest between any two nodes is high if there is a large volume of demand between the two nodes and they are in close proximity to one another. Rings with high community of interest are desirable because they minimize the amount of inter-ring demand. Three of the methods discussed in Section 6.3 decompose the problem along these lines. Three other methods that do not fit these broad categories are also discussed at the end of Section 6.3. The chapter concludes with a summary of the design methods and a comparison of this prior work with the methods developed in Chapters 8 through 11 of this thesis.

6.2 The Single Ring Network Design Problem

This section discusses prior work on the single ring network design problem. In total, we survey six different methods that have been proposed in the literature for several variants of the problem. The first two methods consider the case where a *subset* of nodes are selected to form a ring that maximizes profit or revenue. The other three consider hybrid architectures where the nodes that are not included in the single ring are connected via point-to-point systems.

6.2.1 Gendreau et al.

Gendreau et al. [GLL95] consider the problem of constructing a single ring that connects a subset of nodes in a metropolitan area network. Specifically, the problem is to determine the subset of nodes and the fibre routing between nodes that maximizes the profit derived from the ring subject to a budget constraint. The profit is equal to the sum of revenues between all node pairs served by the ring minus the total fixed cost of all spans on the ring. The revenue derived from each node pair is assumed to be a direct function of the traffic between them. Although the authors considered the problem in the context of a packet-switched ring network, the results are applicable to the equivalent circuit-switched ring network design problem where ring capacity is not a constraint (i.e., the uncapacitated version of the problem). When the revenues are large relative to construction costs, it pays to include all nodes in the ring and the problem is equivalent to the Travelling Salesman Problem (TSP), which is known to be *NP*-complete. Because it is unlikely that large problems can be solved to optimality, the authors propose three greedy constructive heuristics and three post-optimization heuristics. The basic idea behind all three greedy heuristics is to start by choosing the pair of nodes with the maximum profit and then successively enlarging the ring at each iteration by adding the node yielding the maximum increase in profit. The post-optimization procedures are based on similar techniques developed by the authors for the TSP. The heuristics and post-optimization procedures are compared on randomly generated problem instances with up to 400 nodes.

6.2.2 Fink et al.

The same problem addressed by Gendreau et al. is also studied by Fink, Schneidereit and Voβ [FSV98], [FSV99], who develop two greedy heuristics for the problem. The two heuristics are similar to those by Gendreau et al. except one includes a finite look-ahead procedure. They also propose two improvement procedures: one based on Simulated Annealing (SA) and the other on Tabu Search (TS). The initial starting solutions for the SA and TS improvement procedures are obtained from either of the greedy heuristics. The various methods were tested on randomly generated instances of the problem with up to 120 nodes. On average, the results showed that the Tabu Search procedure combined with the greedy heuristic with finite look-ahead provided the best overall results.

6.2.3 Lee, Ro and Tcha

Lee, Ro and Tcha [LRT93] consider the problem of constructing a single ring through a set of hub nodes and linking the hub nodes to a set of non-hub nodes via point-to-point systems. In this case, the hub ring is a unidirectional ring and both the ring and point-to-point systems not restricted to a discrete set of capacities (i.e., uncapacitated). They call this a *ring-star* architecture. The objective of the problem is to minimize the total construction cost of the unidirectional ring and the point-to-point systems. The proposed cost model contains three cost elements: variable flow costs, fixed hub node costs, and fixed span costs. The min-cost solution for this problem must specify the location of the hub ring as well as the hub node to which each non-hub site is connected. Specifying the location of the hub ring involves two inter-related subproblems: (1) determining the subset of candidate hub nodes to include in the ring, and (2) finding the min-cost fibre route that connects the hub nodes in a ring. Both of these subproblems are known to be *NP*-complete. To solve this problem the authors develop an IP formulation for the problem, which is then used to construct a dual-ascent heuristic for finding the near-optimal solution quickly. A heuristic procedure was also developed to provide an upper bound on the solution. Computational experiments were conducted on a variety of randomly generated networks with up to 20 candidate hub nodes and 50 non-hub nodes. Results showed that the gap between the dual-ascent and primal heuristic solutions was less than 12% for all test cases. The worst-case run time for the largest problem was 321 seconds for the dual-ascent procedure running on an HP-9000 workstation.

6.2.4 Xu, Chui and Glover

Xu, Chui and Glover [XCG99] study a logically equivalent problem to that of Lee et al. but in the context of a Digital Data Service (DDS) network. They develop an exact branch-and-cut algorithm and a Tabu Search procedure for the problem. The performance of these methods is tested on a set of randomly generated problems instances with up to 600 nodes. The results show that for smaller test problems of up to 100 nodes, the Tabu Search procedure finds the optimal solution in all cases but requires only a fraction of the time needed by the branch-and cut algorithm. On larger problem instances for which optimal solutions are not available, the TS procedure consistently outperforms the best previous local search methods.

6.2.5 Chamberland and Sansó

A slightly different version of the problem addressed by Xu et al. is also considered by Chamberland and Sansó [ChS97]. In this case, there are capacity constraints on the selected hub sites. The authors present a greedy heuristic and a Tabu Search algorithm and compare the performance of both approaches on problem instances with up to 20 candidate hub sites and 200 non-hub sites. On average, the TS procedure finds solutions that are within 0.18% of the lower bound and in 7 out of the 12 test cases it finds the optimal solution.

6.2.6 Chung et al.

Chung et al. [CKY96] study a variant of the problem in which the non-hub nodes are connected to a single ring using diversely routed 1+1 APS systems. Here the 1+1 APS systems may be connected to either one or two nodes in the ring. The problem involves determining the subset of nodes to include in the ring, the spans used to connect the ring, and the diverse routes used to connect nodes not included in the ring. The authors formulated this problem as an integer program and also develop an alternative heuristic that quickly generates near-optimal results. The heuristic decomposes the problem into three separate subproblems: determining the subset of nodes included in the ring, finding the min-cost ring route that connects the ring nodes, and finding the shortest pairs of span-disjoint routes between non-ring nodes and the ring. The algorithm begins by generating an initial feasible solution and then tries to refine it using several improvement heuristics. A comparison between the heuristic and IP approach shows that results obtained with the heuristic method are within 1% of the optimal solution (obtained using CPLEX) for a number of small random networks. The computation time for the heuristic method is also a small fraction (approx. 1/3000 - 1/10000) of that required for the optimal solution.

6.3 The Multi-Ring Network Design Problem

This section discusses twelve methods proposed in the literature for the multi-ring network design problem. In Section 6.3.1 through Section 6.3.4 we provide an overview of four methods proposed for the span coverage problem. This is followed by a description of three other methods in Sections 6.3.5 through 6.3.7 that consider the single period design problem but do not fall into the categories discussed earlier. In Section 6.3.8 and 6.3.9 we discuss two methods that adopt a hierarchical decomposition of the problem. The final three subsections cover methods that consider the multi-period design problem using a logical ring decomposition.

6.3.1 RingBuilder (Grover et al.)

One of the first methods proposed for the multi-ring network design problem is by Slevinsky, Grover and MacGregor [SGM93], [GSM95], [Sle99]. The authors consider the single-period version of the problem and decompose it into what we now call the *span coverage problem*. The basic design method, known as RingBuilder, uses a greedy heuristic approach that divides the ring design problem into four steps: ring candidate generation, demand routing, candidate ring loading and ring selection. Although several versions of the basic approach have been implemented over the years, we discuss only the most recent versions [GSM95], [Sle99] for the sake of brevity.

In the first step of the approach, a set of candidate rings are generated by finding the cycles within the network graph and instantiating a ring candidate for each combination of cycle and ring technology under consideration. The set of candidate rings is then saved for subsequent use in the candidate ring loading and selection stages. The candidate generation routine also includes a option to limit the maximum number of nodes or the circumference of the candidate rings.

In the second step, the point-to-point demands are routed over the network graph using a shortest-path routing algorithm. Several options are available for routing demands. One option allows all demand between a specific pair of nodes to be handled either as a single entity or separately for each unit of demand (e.g., DS3). Another option permits demand to be distributed as evenly as possible over all shortest routes between the origin and destination nodes. This is referred to as the *k-way splitting* option. The rationale for splitting demands among equally short routes is that it should, on average, distribute demands more uniformly over the underlying network topology. The route length may also be based on either the number of hops (spans) in the route or geographical distance.

After the candidate rings have been generated and demands have been routed, an iterative greedy process is used to construct a feasible solution one ring at a time. At each iteration, each candidate ring is temporarily loaded with demands whose routes intersect the ring. RingBuilder supports two kinds of ring loading for BLSRs: *balance-biased loading* and *capture-biased loading*. In both types of loading, unit demands are sorted in non-increasing order of the number of spans that the demand's route has in common with the cycle of the candidate ring. These demands are then loaded onto the candidate ring in order from longest to shortest. When the capacity on a span is exhausted, the balance-biased loading routine will continue to load demands that intersect the exhausted span but only on segments of the route that are not yet exhausted. In contrast, the capture-biased loading routine will reject demands that cannot be entirely loaded onto the ring. Balance-biased loading improves ring fill at the expense of higher inter-ring transition costs. Capture-biased loading has the opposite effect.

Once all candidate rings have been loaded, the efficiency of the ring candidates is calculated based on the set of demands loaded onto the ring. Two measures of efficiency are proposed by the authors. The first efficiency measure takes the weighted sum of two other metrics known as balance efficiency and capture efficiency [GSM95], as follows:

$$\eta_{BC} = \alpha \cdot \eta_B + (1 - \alpha) \cdot \eta_C, \quad 0 \le \alpha \le 1 \tag{6.1}$$

where η_B is the balance efficiency, η_C is the capture efficiency of the design, and α is called the *balance bias factor*. Balance efficiency (or capacity efficiency) is a measure of how effectively the capacity of the ring cover R is used and is given by

$$\eta_B = \left(\sum_{l=0}^{n} w_l \cdot \lambda_l\right) \Big/ \left(\sum_{l=0}^{n} s_l \cdot \lambda_l + \sum_{l=0}^{n} p_l \cdot \lambda_l\right) \tag{6.2}$$

where w_l is the number of working links on span l, λ_l is the length of span l, s_l is the number of slack or unused links on span l, and p_l is the number of protection links on span l. Balance efficiency is inversely proportional to transmission-related costs (e.g., fibre, ADM common equipment and tributary ports, etc.) because the higher the balance efficiency the less transmission capacity (and equipment) required for a given demand routing pattern. High-balance efficiency is important in long-haul networks where transmission costs outweigh inter-ring transition costs (e.g., DCS common equipment and tributary ports).

Capture efficiency is a measure of how well the demands are contained (i.e., originate and terminate) within the rings and is given by:

$$\eta_C = \frac{(2 \cdot L - T)}{2 \cdot L} \qquad (6.3)$$

where L is the total number of links (or route segments) loaded onto the ring, and T is the total number of inter-ring transitions. Capture efficiency is inversely proportional to inter-ring transition costs. That is, the higher the capture efficiency, the lower the inter-ring interface costs for a given network design problem. High capture efficiency is desirable in metropolitan networks where inter-ring transition costs typically dominate the total cost of the network. Note that by sweeping the balance bias factor, several designs can be generated for comparative purposes and sensitivity testing.

In the most recent version of RingBuilder [Sle99], the ring efficiency is measured by specific progress. *Specific progress* is the ratio of the progress made in completing the design to the cost of constructing the candidate ring. The amount of progress is taken as the sum of the demand-distance product for all demands served by the ring. The construction cost includes the fibre, ADM common equipment, ADM tributary interface cards and inter-ring transition costs. The advantage of the specific progress metric is that it provides a technology-independent means of comparing candidate rings, which makes multi-technology designs possible.

At each iteration after all ring candidates have been evaluated, the ring with the highest efficiency is selected. The demands (or portions thereof) served by this ring are then removed from the pool of unserved demand and the ring loading and ring selection steps are repeated until all demands have been satisfied.

The authors compare the performance of the various program options for several real test networks. For demand routing, the results show that treating each demand unit separately rather than as a bundle during demand routing (and ring loading) reduces design costs by up to 10% in some test cases. The k-way splitting option, on the other hand, does not always result in lower cost designs relative to the single shortest path routing. In metropolitan networks, the costs of designs generated with the shortest-hop routing option are roughly 5% lower than those generated with the shortest-distance option. In contrast, the shortest-distance option provided marginally better results in long-haul networks. The lowest cost designs for single-technology designs are obtained using the weighted balance/capture metric by sweeping the balance bias factor over the interval [0,1]. However, in all cases the specific cost metric provided designs within a few percent of the best solution and only required a single run.

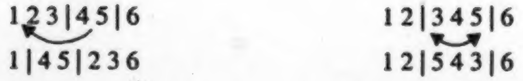
6.3.2 Roberts

The single-period span coverage problem is also studied by Roberts [Rob94a], [Rob94b] who considers the uncapacitated version of the problem for BLSR ring designs only. The author presents a solution technique based on Simulated Annealing (SA). SA is heuristic approach for solving combinatorial optimization problems that is based on a computational analogy to the physical process of annealing in metals. The proposed heuristic is similar to the original version of RingBuilder except the method by which ring candidates are chosen is based on SA rather than a greedy selection process. Because the rings are not restricted to a discrete set of capacities, the author assumes that the objective is to minimize total network redundancy

$$r = \frac{\sum_{r \in R} \sum_{l \in r} p_l}{\sum_{r \in R} \sum_{l \in r} w_l^r} \tag{6.4}$$

Note that this objective function is identical to that of the original version of RingBuilder [GSM93]. The first step in this approach is to find a set of candidate rings for the given network graph *G*. A depth-first search is used for this purpose, as in RingBuilder. Then, the SA algorithm is run using the cycle set as input. The SA algorithm begins by placing the first cycle in the cycle set onto the network graph to form a ring. The capacity of this ring is determined by the working demand assigned to each span of the ring. Once the ring is in place, the working demands are then removed from the spans and the next cycle in the set is placed onto the network. The algorithm continues placing cycles in order until every span that has been assigned working demands is covered by a ring. The subset of cycles placed on the network represents the design solution. Therefore, it is the sequence of the cycle set that uniquely determines the final solution.

Next, the initial solution is modified using a "Generate" function that randomly changes the order of the entire cycle set. The Generate function uses two operations to modify the order of the cycle set: a choose and insert operation, and a reverse segment operation. These operations are illustrated in Fig. 6.4



(a) Choose and insert operation (b) Reverse segment operation

Figure 6.4. Simulated Annealing generate operations.

The choose and insert operation, shown in Figure 6.4(a), changes the original order of the cycle set (i.e., 1 2 3 4 5 6) by randomly selecting a segment of the cycle set and inserting it between two elements not in the segment. The reverse segment operation, on the other hand, changes the order of the cycle set by randomly selecting a segment of the cycle set and reversing its order, as shown in Figure 6.4(b).

The redundancy of the new solution is then computed and compared with that of the current solution (or incumbent). If the redundancy of the new solution is less than that of the incumbent, the new solution is saved. If it is greater than the incumbent, it may still be saved depending on the outcome of a random draw. The decision to accept or reject a higher redundancy solution at iteration k is made by comparing a random number generated from a uniform distribution on the interval [0,1] with the value of

$$e^{[r(k) - r(k')]/c} \tag{6.5}$$

where $r(k')$ is the redundancy of the new solution, $r(k)$ is the redundancy of the current solution, and c is a control constant. If the random number is less than the value returned by Eq. (6.5), the inferior solution is accepted; otherwise it is rejected. As the algorithm progresses, the control constant c decreases, thus lower the probability of accepting a higher cost solution. The SA algorithm continues until no further improvements in cost can be made. The primary advantage of the (SA) approach is that it can escape local minima and more effectively search the solution space to find the global minimum.

Roberts compared the performance of the simulated annealing algorithm with the first version of RingBuilder [SGM93] for a number of test networks. In each case, the entire cycle set was used by both heuristics.The results showed that for relatively small networks (6 to 10 nodes), simulated annealing solutions were as good as or better than those generated by RingBuilder. For larger networks, however, the simulated annealing results were inferior to RingBuilder results and required up to an order of magnitude more run time.

6.3.3 *Eulerian Ring Covers (Gardner et al.)*

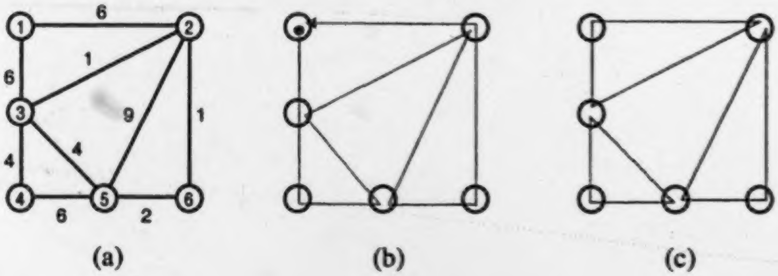
Another design method for the span coverage problem is proposed by Gardner et al. [GHS94]. Here, separate heuristic algorithms are presented for uncapacitated UPSR and BLSR designs. For UPSR designs, it is assumed that cost is a linear function of the working load on each span. That is, the total cost of the network design is

$$\text{cost} \propto \sum_{r \in R} \sum_{l \in r} w_l^r \tag{6.6}$$

where R is the set of rings in the cover and w_l^r is the working load on span l in ring r. In other words, the total cost of the ring cover is proportional to the sum, over all rings r in ring cover R, of the sum of working demand carried by each span in r. Note that the cost per ring, as defined in Eq. (6.6), is not necessarily proportional to the ring's required capacity. This is because the required capacity depends on the sum of demand served by the ring and not the working load on each span. Only when a node-to-adjacent node demand pattern exists is the cost per ring proportional to the required capacity.

In this situation, the problem is similar to the well-known *Chinese Postman Problem* [EdJ73], which involves finding a minimum weight tour that traverses each edge of the graph at least once. It can be shown that if the degree of every vertex in a graph is even, there is an *Euler* tour that traverses each edge in the graph exactly once. An Euler tour is an optimal solution for the Chinese Postman Problem because each edge of the graph is traversed only once. Recall that a graph that contains an Euler tour is called an *Eulerian* graph.

The authors show that if a network graph is Eulerian, the minimum cost ring cover can be obtained by decomposing any Eulerian tour into a ring cover. Such a ring cover is an optimal solution for the present problem because each span is covered only once. An example of a weighted Eulerian graph and an Eulerian tour for this graph are shown in Figures 6.5(a) and 6.5(b), respectively.



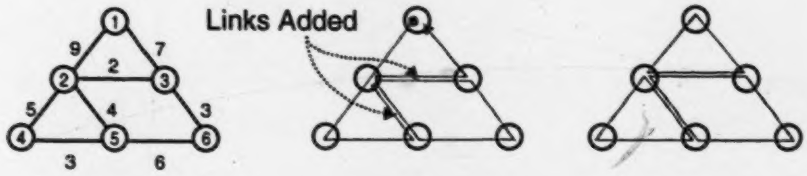
(a) (b) (c)

Figure 6.5. Decomposing an Eulerian graph into cycles.

The Eulerian tour can be decomposed into rings (cycles) by traversing its edges in order and partitioning off a ring whenever a node (vertex) is revisited. The resultant ring cover is shown in Figure 6.5(c). Note that none of the rings in the ring cover overlaps another ring and that each span is covered only once. The total weight of the ring cover in Figure 6.5(c) is 39.

If a graph is not Eulerian, it can be converted to a Eulerian graph by duplicating some of its edges (i.e., by adding parallel edges to the graph). This is equivalent to traversing each (duplicated) edge in the original graph more than once. For the Chinese Postman Problem, the optimal solution is obtained by finding a least cost matching of the vertices of odd degree using shortest paths [EdJ73]. A pair of vertices are *matched* by adding parallel edges along the shortest path between the vertices. Note that the number of vertices of odd degree is always even and the parity (even or odd) of the vertices along the path is unchanged by any such augmentation.

This same basic procedure can also be used to produce a minimum cost ring cover. An example of how a non-Eulerian graph is converted to an Eulerian graph is shown in Figure 6.6.



(a) (b) (c)

Figure 6.6. Converting a non-Eulerian graph to an Eulerian graph.

Note that the graph in Figure 6.6(a) is a non-Eulerian graph because nodes 3 and 5 are of odd degree. The graph is converted to a Eulerian graph by adding parallel edges between 2-3 and 2-5, as shown in Figure 6.6(b). This represents the minimum cost augmentation of the graph for connecting nodes 3 and 5 and making them of even degree. The resultant min-cost ring cover is shown in Fig. 6.6(c). Because spans 2-3 and 2-5 are traversed twice, the total cost of the ring cover is 2 + 4 + 39 =45. Under certain circumstances, however, it may not be possible to decompose the augmented graph into a ring cover because some of the rings may include parallel edges. The authors state the conditions for a feasible ring cover.

For BLSR technology, the cost of the ring cover is given by

$$\text{Cost} = \sum_{r \in R} max(w_l^r) \tag{6.7}$$

Or in words, the cost of the ring cover is proportional to the sum, over all rings *r* in the ring cover *R*, of the maximum working demand carried by any span in *r*. The authors show that the problem of finding the minimum cost ring cover using BLSRs is *NP*-complete by transformation from the undirected Hamiltonian circuit problem, which is known to be *NP*-complete. They present three heuristic methods for finding min-cost ring covers: the *Greedy* method, the *Longest Feasible Ring* method, and the *Maximally Separated Rings* method. All three heuristics use a depth-first search to locate candidate rings (i.e., cycles) in the network graph which are then selected iteratively to construct a ring cover. A priority is also assigned to each node to determine where to start the depth-first search. The heuristics differ only in the manner in which rings (cycles) are selected.

The *Greedy* heuristic starts at the node with the highest priority and selects rings according to the order in which cycles are found using the depth-first search. The procedure continues until all spans in the graph are covered by a ring. The *Longest Feasible Ring* heuristic enumerates all of the cycles in the network graph and selects the one with the maximum number of spans (up to the maximum of sixteen nodes) to be the first ring. This process continues until all spans are covered by a ring. In the *Maximally Separated Rings* heuristic, rings are selected based on the amount of overlap between each candidate ring and the other rings previously placed on the graph, if any. That is, it tries to find a ring cover that minimizes the number of times a span is covered by a ring.

These heuristics are tested on several example networks and the Longest Feasible Ring heuristics and the Maximally Separated Rings heuristic generally perform the best.

6.3.4 Kennington et al.

An integer programming formulation of the single period span coverage problem is proposed by Kennington et al. [KNR97]. Here, the authors consider only uncapacitated BLSR network designs. The authors divide the problem into two steps. The first step involves enumerating a subset of cycles in the network graph. This is done by generating combinations of the fundamental cycle set and eliminating those combinations that are not valid cycles. For small network problems the entire set of cycles is generated; whereas for larger network problems the number of cycles is limited to 1,000 for computational reasons. In the later case, three different methods are proposed for populating the cycle set. The first method populates the cycle set in the order that cycles are found. The other two methods populate the cycle set based on either the redundancy or the cost of the generated cycles.

The second step uses integer programming to find the minimum cost subset of ring candidates that covers all spans. The IP formulation is

Minimize: $$\sum_{\forall r \in S} c_r \cdot \delta_r \ , \tag{6.8}$$

Subject to: $$\sum_{\forall r \in S}^{n} a_{rl} \cdot \delta_r \geq 1 \ , \qquad \text{for } l=1,\ldots,n, \tag{6.9}$$

$$\delta_r = 0 \text{ or } 1 \ , \qquad \forall r \in S \, , \tag{6.10}$$

$$a_{rl} = 0 \text{ or } 1 \ , \qquad \forall r \in S \, , l=1,\ldots,n, \tag{6.11}$$

where c_r is the fixed cost (redundancy) associated with ring candidate *r*, δ_r is 1 if candidate ring *r* is included in the ring cover and 0 otherwise, a_{rl} is 1 if ring *r* covers span *l* and 0 otherwise, and *S* is the set of ring candidates. The redundancy of each ring candidate is calculated from the initial working capacity assigned to each span on the ring. The authors test the performance of this approach on a number of problems from the literature with up to 70 nodes and 175 edges. Optimal solutions are obtained for smaller network examples within about 5 minutes of CPU time. For larger networks with a limited cycle set, the best results are consistently obtained using the cycle generation methods that use cost and redundancy measures to populate the cycle set.

6.3.5 INDT (Doshi et al.)

Integrated Network Design Tools (INDT) is a comprehensive suite of network planning tools developed by Doshi et al. [DDH95], [Dos97] at Bell Labs. In addition to ring design capabilities, INDT also has design modules for mesh and hybrid ring/mesh designs using PDH or SONET/SDH technologies. For the sake of brevity, only the INDT ring network design module is described here. The ring network design module addresses the single-period design problem where discrete capacity UPSR and BLSR rings are used. The authors decompose the multi-ring network design problem into five subproblems or stages: ring generation, ring selection, inter-ring routing, intra-ring routing and load balancing and ring deloading. The design method begins by generating a large set of candidate topological rings. Although the details of the algorithm are not disclosed, some factors that are considered by the ring generation module include constraints on ring circumference, physical route diversity, ring connectivity (for dual ring interconnect) and the mix of intra-ring and inter-ring demand. Next, the ring selection heuristic chooses the best subset of topological rings that minimizes transport costs while satisfying the ring connectivity requirements. The first step in this procedure is to select a subset of topological rings that maximizes the utilization of the underlying WDM and fibre layers. This is done by routing the demand over the underlying mesh network topology. The routing is then optimized by eliminating lightly loaded spans and rerouting demands on the slack capacity on the remaining spans. This results in a reduced subset of topological rings that meet the connectivity requirements. The final set of topological rings is chosen from this subset, taking many of the same factors in ring generation into account.

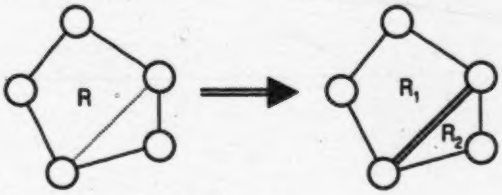
The inter-ring routing stage then assigns the inter-ring demands to topological rings at the origin and destination nodes and determines the end-to-end routing of these demands over the topological ring set. Once this is complete, the total demand on each topological ring is completely specified and the ring sizing problem is solved to determine the routing of demands around the ring. This step is referred to as intra-ring routing and load balancing. Next, the topological rings are partitioned into discrete capacity rings taking care to optimize the placement of ADMs and the assignment of demands to individual rings. The last stage of the method involves removing rings with low utilization and packing the affected demand into the slack capacity in existing rings.

6.3.6 Net-Solver (Gardner et al.)

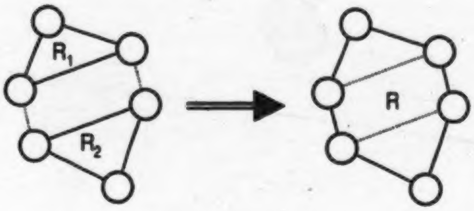
A local search heuristic for the single-period problem is developed by Gardner et al. [GST95]. This method, called Net-Solver, uses a four step iterative approach to generate designs composed of discrete capacity UPSR and BLSR rings. The four steps are ring cover generation, demand routing, costing and ring cover selection. In the first iteration, the user is required to enter an initial ring cover for the network. The authors explain that the initial ring design does not need to be particularly efficient and may be generated manually or by some other simple method (e.g., depth-first search). Once the initial ring cover is specified, the program routes demands over the initial ring design using one of the following routing heuristics: (1) shortest path routing; (2) minimum ring transition routing; or (3) minimum congestion routing.

After all demands have been routed, the total cost of the initial ring cover is computed based on a user-defined cost model. The authors observe that the total cost is generally a function of the length of fibre spans, the number of ADMs placed at nodes and the number of ADM port cards.

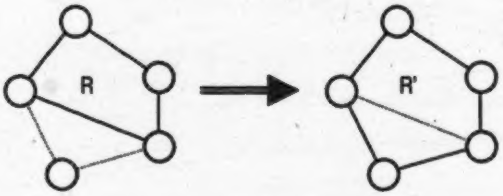
The heuristic then generates a set of alternative solutions using one or more of the operations (or moves) illustrated in Figure 6.7. These include a split operation, merge operation and an enlargements operation.



(a) Split operation



(b) Merge operation



(c) Enlargement operation

Figure 6.7. Net-Solver iteration strategies (adapted from [GST95])

The split operation, Figure 6.7(a), splits an existing ring into two rings. If this strategy is selected, all possible splits are tried for each ring in the ring cover. The merge operation, Figure 6.7(b), merges two node disjoint rings into a single ring via a bridge. A bridge is a pair of spans that connect one ring to another. A similar merge operation also exists for merging two rings that share a single node. The enlargement operation, enlarges an existing ring to include a new node, as shown in Figure 6.7(c). These iteration strategies are then used to generate new ring covers (solutions) by selecting various combinations of existing and modified rings.

For each new ring cover, the program reroutes the existing demands and recomputes the total cost. The ring cover with the lowest cost is then selected as the current design and the iteration procedure begins again. The process repeats itself until no further improvement in cost is achieved, at which point the program terminates. Using the above method, the authors report that equipment costs are often reduced by more than 50% relative to the initial starting solution.

6.3.7 Bortolon et al.

The single period, multi-ring network design problem is also considered by Bortolon et al. [BTR96] for a two-level network consisting of an access and a metropolitan backbone subnetwork. In both subnetworks, the designs may contain BLSR, UPSR or point-to-point systems. The authors divide the problem into four separate phases: hub-node identification, backbone network design, access network design and routing optimization. The purpose of the first step is to identify a subset of nodes in the network to be connected by the backbone network. Here the planner identifies clusters of nodes that have a strong community of interest with the goal of minimizing inter-cluster demand. Note that one node may belong to two different clusters.

The authors formulate the backbone network design subproblem as a fixed charge, multi-commodity network flow problem. The problem inputs include a set of candidate rings, a set of paths for each demand pair and the fixed cost and capacity of each ring candidate. The candidate rings (and point-to-point) systems are input by the user and alternate paths are generated using a shortest path algorithm. A traditional arc-chain (edge-path) formulation [Mur92a] is then used to represent the problem as a mixed integer program. The objective is to minimize the total fixed cost of the UPSR, BLSR and point-to-point systems subject to the following constraint sets: (1) the sum of flows over all paths associated with a demand pair equals the total demand between the nodes, and (2) the cumulative capacity on any link equals or exceeds the total flow over that span. The decision variables include integer variables for the number of each candidate ring and continuous (fractional) variables for the flow over each path. This formulation is similar to the one presented in Section 8.3 except it does not capture the variable cost associated with inter-ring transitions.

The access network design is performed using essentially the same formulation except all nodes (i.e., non-hub and hub nodes) are considered. After the access network design is complete, the last step is routing optimization. The objective of this step is to first minimize capacity utilization and then minimize the number of inter-ring transitions. This problem is also modelled as a linear program that is very similar to the one used in the network design steps except there are only continuous flow variables. The authors present results for a metropolitan area network consisting of 14 hub nodes, 77 non-hub nodes, 65 candidate rings and over 16,000 flow variables. They report that several solutions were obtained for this problem after only a few minutes of processing.

6.3.8 Shi and Fonseka

Shi and Foneska [ShF94], [ShF96], [Shi95] consider the single-period design problem for UPSR and BLSR rings separately. For both problems, they decompose the problem by dividing the design into two or more hierarchical layers and then use a greedy heuristic to generate rings at each hierarchical layer as follows. First, the network nodes at the lowest layer are partitioned into clusters with each node belonging to exactly one cluster. These rings are then interconnected by one or more rings at the next layer in the hierarchy to provide end-to-end routing for inter-ring demands. An example of a two-level hierarchical ring network is shown in Figure 6.8.

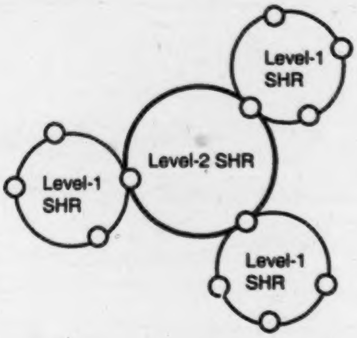


Figure 6.8. An example of a hierarchical ring network.

Depending on the number of nodes in the network, additional layers in the hierarchy may be required to provide end-to-end connectivity. In a the above example, each ring is interconnected to the next higher level ring via a single transit node. The authors also propose an algorithm in [ShF96] for generating designs with two transit nodes (i.e., matched nodes) per ring. Since the algorithms for single and dual transit nodes are similar, we restrict our discussion to the algorithm for dual transit nodes. In this algorithm, cost is assumed to be a linear function of the span distance and the network demands. The cost function is

$$\text{Cost} = \sum_{l \in r} (a \cdot \lambda_l + b \cdot \psi \cdot d_l) \tag{6.12}$$

where a and b are constants, λ_l is the length of span l in ring r, ψ is a capacity-to-demand ratio, and d_l is the demand routed over span l. For UPSR designs, the capacity-to-demand ratio ψ equals one because ring capacity is equal to the sum of all demands on the ring. For BLSR designs, however, ψ is typically less than one because the required ring transmission capacity is generally a fraction of the total demand carried by the ring (due to the bandwidth reuse capability of BLSRs). In this case, the authors suggest setting $\psi = 1$ for the initial design run, calculating the value of ψ in the resultant design, and continuing to iterate in this manner until a min-cost design is achieved.

The algorithm begins by constructing all level-1 rings using the following iterative approach.

First, the node pair with the strongest "community of interest" is identified. The community of interest between two nodes is determined by their physical proximity and mutual demand. The function for selecting the first node pair is

$$\underset{(i,j)}{argmin}(a \cdot \lambda_{ij} - b' \cdot d_{ij}), \qquad \forall ij \in V, i < j \quad (6.13)$$

where, λ_{ij} is the distance between nodes *i* and *j*, $b' = 2b \cdot \Psi \cdot m$ and *m* is the number of nodes in the ring, d_{ij} is the demand between nodes *i* and *j*, and *argmin*() returns an (*i,j*) pair that minimizes the expression inside the brackets, and *V* is the set of all nodes in the network graph. After the first node pair is identified, the nodes and their adjoining span are added to a list that represents a partial ring *r**. The partial ring *r** is then extended by adding another node *k* that has the strongest community of interest with the partial ring. The function for selecting the second and subsequent nodes is

$$\underset{k}{argmin}\left(a \cdot \lambda_{ik} - b' \cdot \sum_{j \in r^*} d_{jk}\right), \qquad \forall i \in \{h,t\}, \forall k \notin r^* \quad (6.14)$$

where *h* is the head and *t* is the tail of the partial ring *r**. Subsequent nodes are added to the partial ring until either the maximum number of nodes in a ring is reached or all nodes in the network graph have been included in a ring. The head and tail of the partial ring are then connected to form a complete ring *r*. Note that in this method, the maximum number of nodes is predefined and is the same for all rings. Next, the algorithm selects two adjacent nodes in *r* to be the transit nodes for that ring. Once again, the two transit nodes are selected on the basis of their community of interest with all other nodes that are not a member of the ring. The function for selecting transit nodes is

$$\underset{(i,j)}{argmin}\left(\sum_{k \notin r} [a \cdot (\lambda_{ik} + \lambda_{jk}) - b' \cdot (d_{ik} + d_{jk})]\right), \qquad \forall ij \in r \quad (6.15)$$

After a ring is complete and the transit nodes have been identified, the algorithm continues assembling level-1 rings until all nodes in the network are covered by a ring.

Level-2 rings are constructed in a similar fashion by connecting the level-1 transit nodes into a ring. If more than one level-2 ring is required to cover all level-1 transit nodes, then at least one additional level will be required. The number of levels required to completely connect the network depends on the number of nodes in the network and the predetermined limit on the number of nodes in a ring. In the final step, shortest path routing is used to route demands and determine the capacity of all rings. The authors do not discuss how BLSR demands are routed within a ring.

To improve the solutions provided by the heuristic algorithm, the authors also investigate the use of simulated annealing (SA) in [ShF96]. The results show that SA achieves a 5-15% reduction in cost over the heuristic algorithm but requires 3 to 4 orders of magnitude longer execution times than the heuristic algorithm.

The performance of the heuristic algorithm is tested in [MoG98b] by comparing the hierarchical design given in [ShF96] with a manual design for the same network. The sample network consisted of sixteen nodes with a randomly-generated demand pattern. The hierarchical design consisted of four level-1 rings and one level-2 ring (all BLSRs), each of which had a maximum of five nodes per ring. In contrast, the manual design consists of a single, sixteen-node BLSR. The results show that the cost of the manual design is 18-39% lower than the hierarchical design, depending on the ratio of distance to demand costs.

6.3.9 Goldschmidt, Laugier and Olinick

The problem of designing a two-level hierarchical ring network is also studied by Goldschmidt, Laugier and Olinick [GLO98]. In this case, however, only UPSRs are considered and all rings are assumed to have the same capacity. Here the problem is divided into two subproblems. The first subproblem involves partitioning the set of network nodes into logical level-1 rings or clusters that satisfy the ring capacity constraint. The second subproblem is then to create a physical ring from each logical ring by finding the shortest fibre route that connects the nodes in the logical ring. Solution techniques originally developed for the Travelling Salesman Problem are used to solve the latter subproblem.

The authors consider two variants of the subproblem of creating the logical rings. In one of these, the objective is simply to minimize the number of logical rings in the design. The authors develop an Integer Programming formulation for this variant and present two heuristic solution methods: an edge-based heuristic and a cut-based heuristic. Both heuristics start by assigning each node to its own ring. The edge-based heuristic examines the demand pairs in decreasing (non-increasing) order of size. If the end nodes of the demand pair are in different rings, it merges the two rings subject to the ring capacity constraint. In the cut-based heuristic, the two rings with the maximum inter-ring demand are merged at each iteration. The authors show that the maximum number of rings placed by any heuristic is at most twice that of the optimal solution. Note, however, that neither of these heuristics guarantees that a feasible solution will be found because the capacity of the level-2 ring is not considered.

In the other variant of the logical ring subproblem, the maximum number of rings *K* is specified in advance and the objective is to minimize the demand served by the level-2 ring. The authors show that the problem is *NP*-complete by reduction from the Graph Bisection Problem. They also formulate this problem as an Integer Programming problem and present a node-based heuristic solution method. In the node-based heuristic, the user starts by selecting the number of rings to construct. One node is then randomly assigned to each ring. The remaining nodes are assigned one by one. At each iteration, the heuristic selects the ring with the greatest slack capacity and adds to this ring the unassigned node that has the greatest amount of demand in common with the current ring. Using a binary search, this method can also be used to find the optimal value of *K* that minimizes the number of rings. Again, this heuristic does not guarantee that a feasible solution will be found.

The three heuristics are tested on eight random networks with up to 50 nodes as well as two real-world networks. The results show that the node-based heuristics found an optimal solutions for seven of eleven feasible problems instances but failed to find a feasible solution in two problem instances.

6.3.10 Strategic Options (Wasem, Wu and Cardwell)

Some of the earliest work on the multi-ring network design problem is by Wasem, Wu and Cardwell [WuC91], [Was91a], [Was91b], [WWC94] at Bellcore. This work resulted in a design method known as Strategic Options, one of three methods surveyed here that models a multi-period planning environment. It also considers network designs with a mix of UPSR, BLSR and diversely-routed 1+1 APS systems. A key assumption of Strategic Options is that the clusters of nodes that must be connected in a ring are specified in advance by the planner. Presumably, these *network clusters* are chosen based on either community of interest or the existing network architecture. In addition, for each network cluster, the hub node or nodes are already predefined. The objective is to minimize the total (discounted) cost of all UPSR, BLSR and diversely-routed 1+1 APS systems deployed over the planning period.

Strategic Options divides the optimization problem into three separate steps: demand bundling, ring selection, and topology optimization. The purpose of the demand bundling step is to determine the most efficient strategy for grooming DS1 demands into STS-1 containers. Two bundling alternatives are considered for each demand pair: direct routing and hubbing. In direct routing, the DS1 demand is converted into an equivalent number of STS-1 demands between the O-D pair. In the hubbing alternative, the DS1 demand at each end is routed (in STS-1 containers) to a pre-defined hub node first and then between the hub nodes serving the O-D pair. The decision to use either direct routing or hubbing may be based on either percentage fill or cost, as specified by the user. When the decision is based on percentage fill, a direct STS-1 is used whenever the demand (measured in DS1s) exceeds a given threshold. Otherwise, the demand is routed at each end to a designated hub node and aggregated with other demands. When the decision is based on cost, the average cost per DS1 is used to compare the direct routing and hubbing alternatives and the alternative with the lowest average cost is selected. The average cost for the hubbing alternative is calculated assuming 100 percent fill, whereas the actual cost is used for the direct routing alternative. The decision to use hubbing or direct routing is evaluated once for every planning period. It is assumed, however, that once hubbing has been chosen in any period it will continue to be used until the end of the planning horizon. If direct routing is chosen, however, it will be used until the capacity is exhausted, at which time both options will be evaluated again. Figure 6.9 shows an example of different sequences of demand bundling choices and their impact on total cost.

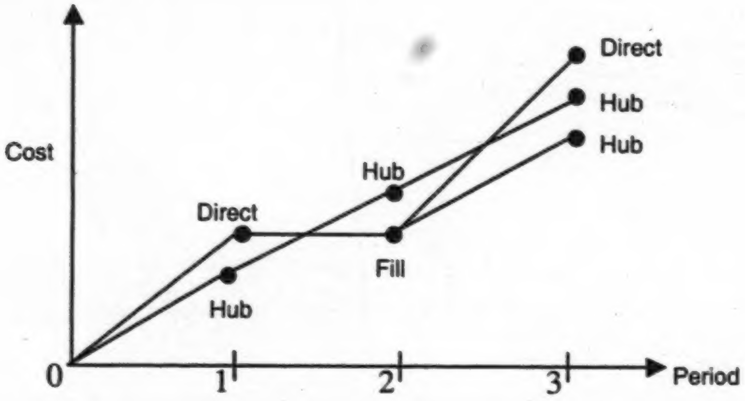


Figure 6.9. Strategic Options: An example of a multi-period demand bundling choices (adapted from [WuC91]).

Dynamic programming is used to find the sequence of demand bundling decisions with the lowest total cost. Note that the result at this stage is just the demand matrix that other methods start with. In a sense, this preprocessing step to determine the demand matrix could be used with any other method.

After demand bundling is complete, the ring selection step compares the cost of rings and diversely routed 1+1 APS systems for each network cluster. To make this comparison, the ring selection heuristic identifies all logical rings that contain the hub node (or nodes if dual homing is used). The heuristic does this by first finding all of the cycles in the network graph containing the hub node(s). Then for each cycle, it lists all possible combinations of active nodes (i.e., nodes equipped with an ADM). An example of thi. procedure is illustrated using the network graph in Figure 6.10.

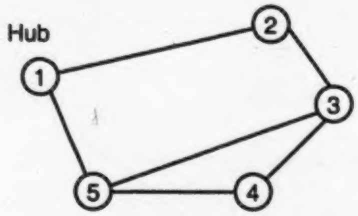
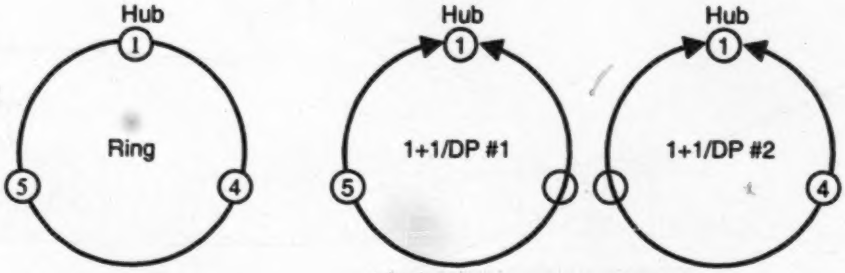


Figure 6.10. Strategic Options: An example of ring selection (from [WuC91]).

In this figure, there are two cycles: 12345 and 1235. In cycle 12345, the logical rings are 123, 124, 125, 134, 135, 145, 1234, 1245, 1345, 12345. For each logical ring, dynamic programming is used to determine the optimal choice of ring (i.e., either UPSR, BLSR, and 1+1 APS) to deploy in each period. An example of the design alternatives for candidate ring 145 is shown in Figure 6.11.



(a) Ring alternative

(b) 1+1/DP alternative

Figure 6.11. Strategic Options: An example of the design alternatives for a potential ring.

The above procedure assumes that once an 1+1 APS system is placed, 1+1 APS systems will be used until the end of the planning horizon. On the other hand, if a ring is placed it is used until exhausted at which time all options are evaluated again. Figure 6.12 shows an example of different sequences of ring choices. The logical ring with the lowest total cost is selected and any other candidate rings containing the same non-hub nodes are then deleted from the list of candidate rings. For example, if ring 145 is chosen, then ring candidates 124, 125, 134, 135, 145, 1234, 1245, 1345, 12345 are removed. Only ring candidate 123 remains for future consideration. This process repeats until all nodes have been covered by a cycle.

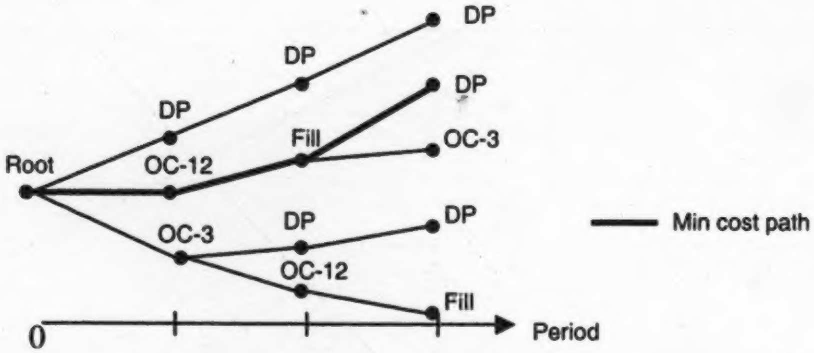


Figure 6.12. Strategic Options: An example of a multi-period multiplex cost algorithm (adapted from [WuC91]).

After all logical rings have been selected, a fibre routing heuristic is used to find a span-disjoint fibre route to connect the active ring nodes. The fibre routing algorithm uses a two-stage approach. The first stage attempts to build a ring by finding the two shortest span-disjoint paths (i.e., the shortest cycle) between two active nodes: an arbitrary active node (the hub) and the active node that is farthest from the hub. If the shortest cycle does not contain all active nodes, a more in-depth search is performed. This second search finds all shortest paths between active nodes and concatenates combinations of these paths together to build a ring that includes all active nodes. This process is repeated until all nodes in the cluster have been covered.

Once the ring selection step is complete, a topology optimization heuristic explores alternative fibre routing for all rings and APS systems with the goal of minimizing the total topology-related costs of the network. Topology costs include such items as route mileage (installation), fibre (material and splicing) and regenerator costs. A greedy heuristic is used here to construct a feasible two-connected sub-network within the network. Next, spans contained in the initial rings (from the ring selection step), but not in the two-connected sub-network, are added to the current solution to ensure that the topology contains the initial ring routings. Two improvement heuristics are then used to improve the solution by selectively adding and removing spans to minimize the overall topology costs.

6.3.11 SONET Toolkit (Cosares et al.)

SONET Toolkit [CSW92], [CDS95] is a multi-period planning tool for designing hybrid SONET networks using survivable rings (e.g., UPSR and BLSR) and other SONET architectures. The toolkit is based, in part, on Strategic Options and other transport planning tools developed at Bellcore. The ring design module divides the design problem into four steps.

The design method begins by selecting a set of hub nodes for aggregating demands that do not require protection or that are too small to warrant direct connections. This is done by comparing the cost of adding new hubs to the cost of transporting the affected demands to smaller set of hub nodes. Once the hub locations are determined and the demand matrix is modified accordingly, the network nodes are partitioned into clusters (logical rings) using a greedy heuristic. The heuristic starts by selecting a pair of nodes that share a significant amount of demand and physically close to one another (i.e., with a strong community of interest). Additional nodes are then added one at a time by evaluating the community of interest between the candidate nodes and the current cluster. This process continues until user-defined limits are met. Next, the fibre routing heuristic used in Strategic Options (Section 6.3.10) is used to find a suitable cycle for connecting the nodes to form a ring. If a suitable cycle is found, a ring is created and included in the final design. Otherwise, the cluster is rejected. Once a ring is identified, the algorithm determines the type and number of ADMs required for the proposed ring. The cost of the ring is then compared to an equivalent configuration using diversely-routed 1+1 APS systems (like Strategic Options) or a combination of point-to-point systems and a ring. The algorithm then selects the appropriate architectures. The final step determines the appropriate size of the rings and 1+1 APS systems that were selected and the corresponding multiplexer equipment, fibre cable and other support structures.

6.3.12 Cox et al.

The multi-period, multi-ring network design problem is also considered by Cox, Yuping, Tegan and Lu [CQT96]. The authors decompose the problem into three main steps: logical ring design, physical ring design and routing optimization. The purpose of the logical ring design step is to find the set of logical rings to deploy in each period such that all demands are served at the minimum total (discounted) cost. Each logical ring is specified by the subset of nodes that are to be equipped with ADMs and connected to form a ring. The authors formulate this subproblem as an Integer Program. The objective is to minimize the (discounted) fixed ADM costs and electrical and optical inter-ring transition costs subject to constraints on ring capacity, the number of ADMs per ring and transit node capacity. The decision variables include, for each time period, the nodes assigned to each logical ring and the intra-ring and inter-ring demand flows. A heuristic procedure is also presented for networks with a large number of nodes. The heuristic begins by assigning the two nodes from the largest unserved demand pair to a logical ring. The ring is then enlarged by adding a neighbouring node with the greatest unserved demand in common with the nodes on the ring. Additional nodes are added until the ring capacity is exhausted. This process repeats until all demand is served.

After the logical rings are defined, physical fibre links are assigned to connect the nodes in the logical rings. This physical ring design step is performed manually by the user who must determine the most efficient placement of physical rings.

The last step in the design procedure determines the (near) optimal routing of demands through the established rings. The authors formulate this subproblem as an Integer Program, where the objective is to maximize the demand served minus the inter-ring transition costs. Aside from the objective function, the formulation is almost identical to that for the logical ring design subproblem. A heuristic procedure is not proposed for the routing optimization step.

6.4 Summary

A summary of the attributes of the design methods described in the preceding sections is presented in Table 6.1. In Chapters 8 through 11, we develop several alternative methods for the multi-ring network design problem. All of these methods consider the single-period version of the problem where the design consists of a set of discrete-capacity rings. With the exception of one method, they may all be used (or modified) to generate designs containing both UPSR and BLSR rings. At a high-level, most of these methods are similar in some respects to those surveyed here but differ significantly in the details. For example, in Chapter 8 we develop a greedy heuristic approach based on the RingBuilder framework to serve as a benchmark for the other design results. In the process, we have implemented a number of enhancements and improvements to the basic approach. These are discussed in detail in Chapter 8.

In Chapter 10, we develop three IP formulations for the multi-ring network design problem. One of these is similar to the IP proposed by Kennington except it considers the capacitated version of the problem. The second IP models the problem as a multi-commodity flow problem using the arc-chain formulation. This is similar to the formulation proposed by Bortolon et al. except the cost of inter-ring transitions is modelled in the current formulation. The third IP formulation is a novel approach that has not been considered previously in the literature. In Chapter 11, we develop local search procedure based on Tabu Search. To the best of our knowledge, this is the first application of Tabu Search to the version of the multi-ring network design problem discussed here.

Table 6.1: Comparison of Prior Work on Multi-Ring Network Design

Method	Optimization Technique	Decomposition	Periods	Capacitated	Technologies
RingBuilder	Heuristic	Span Coverage	Single	Y	UPSR+BLSR
Roberts	SA	Span Coverage	Single	N	BLSR
Eulerian Rings	Heuristic	Span Coverage	Single	N	UPSR,BLSR
Kennington et al.	IP	Span Coverage	Single	N	BLSR
INDT	Heuristic	Topological Rings	Single	Y	USPR+BLSR
Net-Solver	Heuristic	Local Search	Single	Y	UPSR,BLSR
Bortolon et al.	MIP	MCF Relaxation	Single	Y	UPSR+BLSR
Shi & Fonseka	Heuristic	Logical Rings	Single	Y	UPSR,BLSR
Goldschmidt et al.	IP/Heuristic	Logical Rings	Single	Y	UPSR
Strategic Options	Heuristic	Logical Rings	Multiple	Y	UPSR+BLSR
SONET Toolkit	Heuristic	Logical Rings	Multiple	Y	UPSR+BLSR
Cox et al.	IP/Heuristic	Logical Rings	Multiple	Y	UPSR

7. Research Methodology

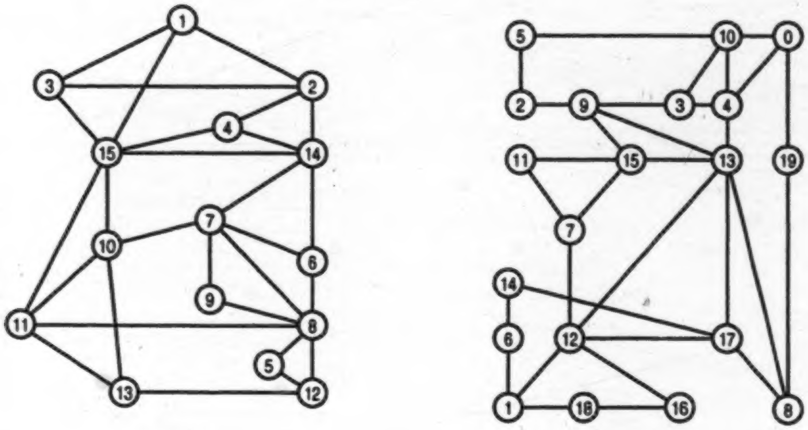
7.1 Introduction

This thesis introduces and characterizes several new aspects and approaches to the multi-ring design. This chapter defines our methodology and approach to the research. For instance, how will we judge when a new idea leads to an improvement or not? How can we know how close a design result is to the optimum? This chapter presents the test networks and experimental methods used to assess the performance of the design methods developed later. To compare the performance of these design methods we conduct a series of controlled tests. The main set of tests are conducted using four test networks under three different technology scenarios. By testing these methods across a wide range of problem instances, we hope to gain a general idea of how well each method performs and under what circumstances. For each design method, we also conduct several other tests to assess the specific performance of individual methods for a range of parameter settings. These specific tests are performed over a narrower range of problem instances using the same problem data and are described in subsequent chapters. The following subsections describe in detail the test networks, modeling assumptions and test cases used to evaluate design method performance. We also describe the metrics and evaluation procedures used to assess the relative and absolute performance of each design method.

7.2 Test Networks

To ensure that the findings of the work are representative of real problems, the test networks are based on data from actual transport networks. In total, four test networks were used to assess the performance of the proposed design methods. The size of the test networks range from 15 nodes, 28 spans for the smallest network up to 43 nodes, 84 spans for the largest. Two of the networks are based on data from metropolitan area networks. The topologies of the metropolitan area networks are shown in Figure 7.1. The network in Fig. 7.1(a), Net15, is from a study published in [Bel93] and contains 15 nodes and 28 spans. The network in Fig. 7.1(b), Net 20, is from [SGM93] and contains 20 nodes and 31 spans.

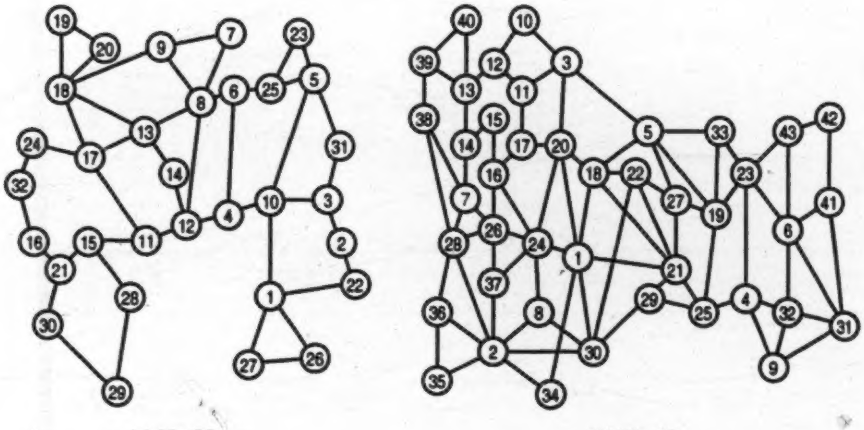
The other two test networks are based on data from long-haul transport networks. The topologies of these networks are shown in Figure 7.2. The network in Fig. 7.2(a), Net32, contains 32 nodes and 45 spans and is from an actual network study. The network shown in Fig. 7.2(b), Net45, contains 45 nodes and 83 spans and is based on data from a continental-scale network. The source for these networks is not disclosed here to respect an agreement with an industrial collaborator. A summary of the topology statistics for all four networks appears in Table 7.1.


1
3
2
4
15
14
7
10
6
9
11
8
5
13
12
5
10
0
2
9
3
4
11
15
13
19
7
14
6
12
17
1
18
16
8

(a) Net15

(b) Net20

Figure 7.1. Topology of the metropolitan test networks


19
20
9
7
23
18
8
6
25
5
13
24
17
31
14
32
12
4
10
3
16
15
11
21
2
22
28
1
30
27
26
29
40
10
39
12
3
13
11
38
15
5
33
43
42
14
17
20
18
22
16
23
27
19
41
7
26
28
24
1
6
21
37
29
25
4
32
31
36
8
2
30
9
35
34

(a) Net32

(b) Net43

Figure 7.2. Topology of the long-haul test networks

As shown in Table 7.1, the average nodal degree (i.e., the average number of spans incident on each node) of the test networks ranges from 2.8 for Net32 up to 3.9 for Net43. The average span length of the metropolitan areas networks is only 4.0 and 6.5 kilometers, for Net15 and Net20, respectively. As a result, the majority of the cost in metropolitan networks is due to terminal equipment costs rather than distance-dependent costs such as fibre material and installation and regenerators. In contrast, the average span length in long-haul networks is in the order of a couple hundred kilometers and, therefore, distance is the main cost driver. Note this is why long-haul networks generally have higher average nodal degrees (i.e., connectivity), which keeps distance-related costs to a minimum. With the advent dense WDM systems, however, total network costs are becoming less sensitive to distance in long-haul networks.

Table 7.1: Test Network Topology Statistics

Network	Type	Nodes	Spans	Avg. Nodal Degree	Avg. Span Length (km)
Net15	Metro	15	28	3.7	4.0
Net20	Metro	20	31	3.1	6.5
Net32	Long-haul	32	45	2.8	352.0
Net43	Long-haul	43	84	3.9	153.7

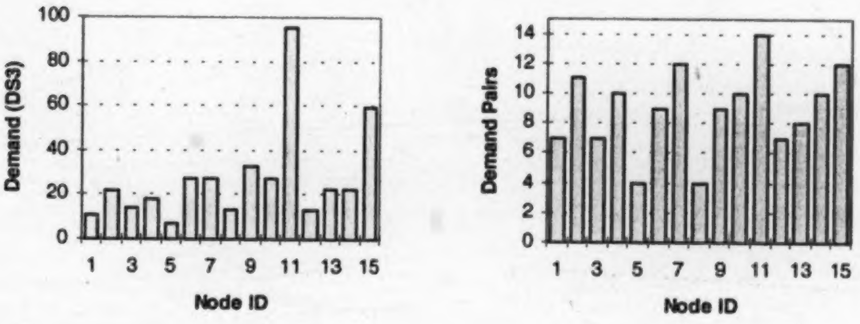
Table 7.2 provides a summary of the demand statistics for all four networks. In some cases, the original network data was given in DS1s but for testing purposes all demands are scaled to DS3 units. The demand patterns for the metropolitan test networks are also listed in Appendix C.

Table 7.2: Test Network Demand Statistics

Network	Demand (in DS3s)			O-D Pairs		
	Total	Avg./Node	Peak-to-Avg	Total	Avg./Node	Peak-to-Avg.
Net15	206	13.7	3.5	67	3.08	1.57
Net20	348	17.4	6.1	126	2.76	1.35
Net32	354	11.1	7.2	72	4.92	5.56
Net43	2525	58.7	1.2	903	2.80	1.00

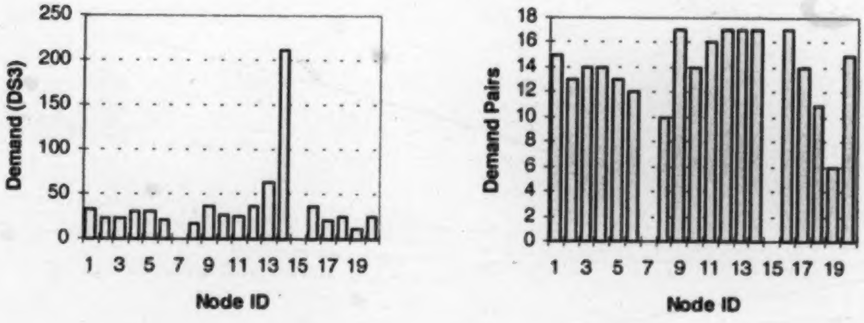
In Table 7.2, the Peak-to-Average columns are calculated by dividing the maximum demand (number of O-D pairs) for any node in the network by the average demand (number of O-D pairs) over all nodes. These peak-to-average values provide additional insight into the demand pattern for each network. For example, a high O-D pairs peak-to-average value indicates that at least one node has a higher than average number of demand pairs, as would be expected in a hubbed demand pattern. A low value, on the other hand, indicates that demand pairs are more evenly distributed, as in a mesh demand pattern. Similarly, the demand peak-to-average value gives an indication of the concentration of demand among the nodes. For example, in Net15 we see that at least one node originates/terminates three and a half times as much demand as the average node. The O-D pairs peak-to-average value, one the other hand, is moderately low. This suggests that while the demand pattern in Net15 is primarily mesh-like, there is at least one primary node (e.g., a toll-office) that originates/ terminates a large majority of the overall demand. A similar pattern is observed in the case of Net20, except that the demand peak-to-average value is even higher.

These observations are supported by Figures 7.3 and 7.4, which show the distribution of demand and demand pairs for networks Net15 and Net20, respectively. The histograms in Figures 7.3(a) and 7.4(a) shows the total quantity of demand (in DS3s) originating from each node in networks Net15 and Net20, respectively.



(a) (b)

Figure 7.3. Demand distribution for Net15: (a) total demand per node (in DS3s), (b) total number of demand pairs per node.

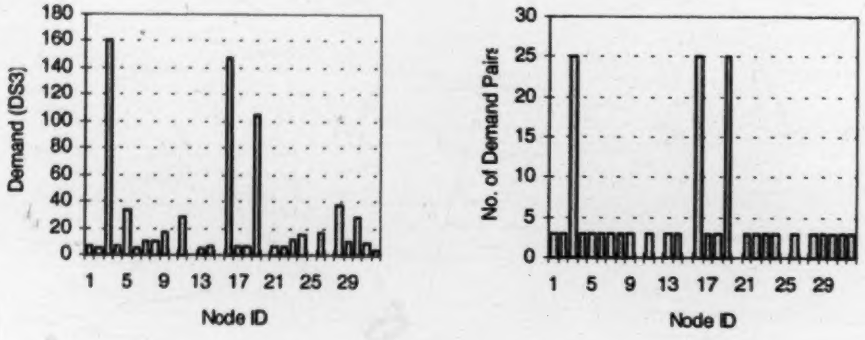


(a) (b)

Figure 7.4. Demand distribution for Net20: (a) total demand per node (in DS3s), (b) total number of demand pairs per node.

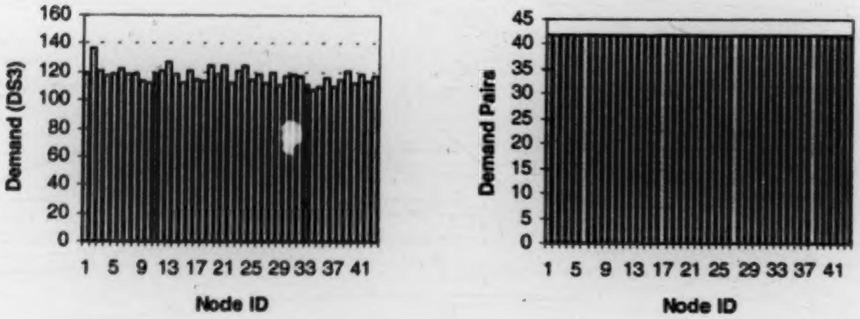
Similarly, the histograms in Figures 7.3(b) and 7.4(b) shows the total number of demand pairs originating at each node in networks Net15 and Net20, respectively. These figures show that there is one node in both Net15 and Net20 that handles a large proportion of the overall demand. This demand pattern is characteristic of metro area networks, where one or two nodes serve as a toll office or gateway to a long-haul network.

The demand and demand pair distributions for Net32 and Net43 are shown in Figures 7.5 and 7.6, respectively.



(a) (b)

Figure 7.5. Demand distribution for Net32: (a) total demand per node (in DS3s), (b) total number of demand pairs per node.



(a) (b)

Figure 7.6. Demand distribution for Net43: (a) total demand per node (in DS3s), (b) total number of demand pairs per node.

In Figure 7.5, we see that all of the demand in Net32 originates/terminates at one of three hub nodes. This demand pattern is not typically found in long-haul transport networks and suggests that perhaps the network is designed for a specific application (e.g., Internet backbone). In contrast, Figure 7.6 shows that Net43 has a uniform mesh demand pattern. That is, there is demand between every pair of nodes in the network. The total demand originating/terminating at each node is also very evenly distributed. These distributions give rise to the low peak-to-average values in Table 7.1.

7.3 Modeling Assumptions

This section describes the basic modeling assumptions that were adopted for the main test cases described in Section 7.4. Unless otherwise noted, these modeling assumptions also apply to any specific tests conducted on each of the proposed design methods.

All tests conducted herein assume a single period (or static) planning environment, in keeping with the demand data for the test networks. For comparative purposes, the demands are routed over the shortest geographical distance in all test cases. We also assume that there are no restrictions on demand routing aside from those due to ring and DCS capacity constraints. That is, demands may be split on an integer basis over any number of routes between their respective origin and destination nodes. The demand on any given route may also be carried by more than one ring and, in the case of a BLSR, may be further split between the two directions around the ring as required. All nodes are equipped with a DCS for provisioning speed and flexibility. Demands that transit from one ring to another are routed through the DCS at the transit node. Therefore, a DS3 demand consumes two ADM add-drop ports and two DCS add-drop ports per inter-ring transition. At origin and destination nodes, however, demands are terminated directly on the client layer equipment rather than passing through the co-located DCS. In all cases, there are no constraints on total DCS capacity. It is also assumed that there are no constraints on fiber capacity on any network spans.

The subsequent subsections describe the ring types and cost model used in these studies.

7.3.1 Ring Technologies

To limit the number of tests performed in this study, we consider network designs comprised of BLSRs only, even though most of the design methods developed in the subsequent chapters support both BLSRs and UPSRs. Table 7.3 lists the attributes of the BLSR technologies used in the test cases. In all test cases, 4-fibre BLSRs are considered operating at either OC-12 (4B12), OC-48 (4B48) or OC-192 (4B192) line rates, which are typical of commercially available SONET equipment. These line capacities (or multiplexing ratios) are also representative of current WDM ring technology. In all cases, there are no constraints on the number of demands added/dropped at each ADM other than those implied by the ring capacity. For example, the maximum add-drop capacity of the 4B12 and 4B48 rings are 24 and 96 DS3s, respectively. There are also no limits on the number of nodes per ring. However, the maximum number of ADMs in any ring is limited to 16, in keeping with SONET standards [Bel95a], [Bel95b].

Table 7.3: Ring Technologies

Name	Fibres	Type	Line Rate	Add-Drop Capacity	Max. # ADMs	Max. Circum. (km)	Regenerator Spacing (km)
4B12	4	BLSR	OC-12	24 DS3s	16	4,000	80
4B48	4	BLSR	OC-48	96 DS3s	16	4,000	80
4B192	4	BLSR	OC-192	384 DS3s	16	4,000	80

Unless otherwise noted, the maximum circumference is also limited to 4,000 km to meet the protection switching times specified in the SONET standards. In some test cases, however, it is necessary to restrict the ring circumference further to limit the number of ring candidates under consideration. The regenerator spacing for all ring technologies is 80 kilometres. In practice, optical amplifiers are sometimes used to increase regenerator spacing but, for simplicity, we assume that regenerators only are used to meet transmission link budgets on optical power, dispersion and other transmission impairments. It is also assumed that regenerators are used at all glassthrough nodes.

7.3.2 Cost Model

For comparative purposes, the total design cost is modelled using a fixed plus variable cost model. The fixed costs are those costs that must be incurred before any demand can be served. These costs include the cost of common ADM equipment, regenerators and fibre facilities. Variable costs are costs that vary in proportion to the amount of demand served. These costs include the cost of ADM and DCS add-drop interfaces. Although actual equipment and facility costing may be more elaborate (e.g., 2-stage multiplexing), this model is sufficiently accurate for comparison purposes and little would be gained in terms of solution fidelity by using a more complicated (and less tractable) cost model. The relative costs used in this study for ADM common equipment, add/drop interfaces and other network elements and facilities are shown in Table 7.4. These values are representative of typical of transport network equipment and facility costs, however, actual planning costs may vary substantially depending on volume discounts, incentive pricing and technology lifecycle factors.

Table 7.4: Equipment and Facility Costs

Ring Technology	ADM Common Equipment	ADM Add-Drop interface	DCS Add-Drop Interface	Regenerator	Fibre (pair-km)
4B12	1X	0.025X	0.025X	0.2X	0.005X
4B48	2X	0.025X	0.025X	0.2X	0.005X
4B192	4X	0.025X	0.025X	0.2X	0.05X

Based on this cost model and the preceding assumptions, the total cost of a network design using 4B48 ring technology, for example, is given by:

$$\textit{total cost} = (2A + 0.05D + 0.1T + 0.2R + 0.01F) \cdot X \tag{7.1}$$

where A is the number of ADMs, D is the total demand, T is the number of inter-ring transitions and F is the total fibre mileage. Note the second term on the right-hand side of Eq. (7.1) represents the total cost of adding/dropping all demands at their respective origin and destination (O-D) nodes, which requires two ADM add/drop interfaces per unit of demand. The third term is the total cost of all inter-ring transitions, each of which requires two ADM add/drop interfaces and two DCS add/drop interfaces.

7.4 Test Cases

For the main set of results, three different ring technology scenarios are considered for each of the four test networks. These include two single-technology scenarios and one multi-technology scenario per test network. Table 7.5 lists the combinations of test network and technology scenario for each of the main test cases.

Table 7.5: Main Test Cases

Metro-Area Networks			Long-Haul Networks		
Test Case	Network	Technologies	Test Case	Network	Technologies
1	Net15	4B12	7	Net32	4B48
2	Net15	4B48	8	Net32	4B192
3	Net15	4B12, 4B48	9	Net32	4B48, 4B192
4	Net20	4B12	10	Net43	4B48
5	Net20	4B48	11	Net43	4B192
6	Net20	4B12, 4B48	12	Net43	4B48, 4B192

For the two metro area networks, Net15 and Net20, both single and multi-technology designs are considered using 4B12 and 4B48 ring technologies. For the long-haul networks, Net32 and Net43, the 4B48 and 4B192 ring technologies are considered.

For each combination of test network and technology scenario, nine network designs were generated using the design methods developed in Chapters 8 through 11. These include the greedy heuristic algorithm described in Chapter 8, the three mathematical programming approaches outlined in Chapter 10 and the Tabu Search algorithm discussed in Chapter 11. In total some 108 network designs were produced in the main set of test results. Numerous other tests were also conducted for each design method to assess the impact of various parameter settings and other method-specific considerations. These test cases are described in their respective chapters.

7.5 Method of Analysis

7.5.1 Performance Metrics

The two primary metrics used to compare the performance of the design methods are total design cost and runtime. The total design cost for each test case is calculated using the cost model in Section 7.3.2. Because the mathematical programming formulations described in Chapter 10 do not model all details of the network design, it is not possible to make direct comparisons using their objective values. Therefore, the solutions generated by these methods are first completed before calculating the total design cost. The procedure used to complete these designs is described in detail in Chapter 11.

The runtime results for the mathematical programming methods are recorded directly from the commercial optimization software used for solving problem instances. These runtimes represent the actual CPU time consumed by the optimization software. These tests were run on a Sun UltraSparc 450 equipped with 512 MBytes of RAM and four processors, each operating at 250 MHz.

For the greedy heuristic and Tabu Search algorithms, however, the CPU time was not directly available so the user (or clock) time is recorded instead for each problem instance. This is because these design methods were written in the Java programming language, which does not have access to the actual CPU time usage. Because the user time is highly dependent on the current load on the computer, these test cases were run on a dedicated ATX-class personal computer equipped with 256 MBytes of RAM and an AMD Athalon processor operating at 750 MHz. Thus, the runtime results provide only a rough comparison of the runtime performance between the mathematical programming and heuristic design methods.

In addition to total design cost and runtime, several other design statistics were recorded for each test case. These include the following:

(a) Number of rings.

(b) Number of ADMs.

(c) Number of regenerators.

(d) Total fibre mileage.

(e) Average ring utilization (or fill).

(f) Number of inter-ring transitions.

(g) Average number of hops per demand.

(h) Number of unused network spans.

(i) Total working capacity (in DS3-hops).

(j) Total capacity (in DS3-hops).

7.6 Performance Evaluation

Three ways were used to estimate the performance of the design methods: empirical testing, a lower bounding procedure and statistical inference.

7.6.1 Empirical Testing

The most direct means of assessing the performance of each method is by comparing the results from all methods. Ideally, empirical testing would be performed over a wide range of problem instances to measure the overall performance and to determine the circumstances where one method performs better than another. Given the large size of the parameter space and the range of design methods developed here, it is not feasible to comprehensively test all values of the problem parameters across a large sample of problem instances. Instead, we focus on the four representative test networks and conduct a series of tests for each method across a relevant range of the parameter space. While not conclusive, these results show broad performance characteristics of the design methods developed here.

7.6.2 Lower Bounding Procedure

Because none of the design methods actually guarantees optimality for the complete problem, some indication of their absolute performance relative to the optimal solution is clearly of interest. To address this question we adopt two approaches: a lower bounding procedure (discussed in this section), and statistical inference (discussed in Section 7.6.3). For some heuristics it is possible to analyse their operation and derive bounds on worst-case and/or average performance. Given the complexity of the heuristics developed here, however, it is unlikely that bounds can be found analytically. It has been shown, for example, that Tabu Search and other local search methods have no performance bounds for the TSP, even in exponential time [Ree93].

Lower bounds on a design consisting entirely of BLSR rings, however, can be obtained by some conditioning arguments. For example, a lower bound on transmission capacity can be derived based on the necessary (but not sufficient) condition that the cumulative capacity of all rings incident on a span (*i,j*) equals or exceeds its working load w_{ij}. If the ring capacity (or modularity) is denoted by m, then the number Z_{ij} of ring modules on span (*i,j*) must satisfy the following inequality:

$$Z_{ij} \geq \left\lceil \frac{w_{ij}}{m} \right\rceil \qquad \forall (i,j) \in S \qquad (7.2)$$

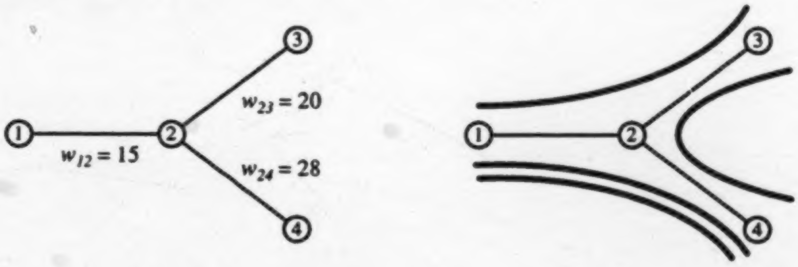
A *ring module* represents the modular capacity of a ring on a single span. Thus, a lower bound on the total transmission capacity is given by

$$\sum_{\forall ij \in S} \left\lceil \frac{w_{ij}}{m} \right\rceil \qquad (7.3)$$

where *S* is the set of spans in the network graph. This lower bound can be refined further by observing that each ring that passes through a node covers exactly two incident spans. Therefore, the total number of ring modules on all spans incident on the same node must be even. This is called the *ring parity condition* and can be expressed mathematically as:

$$\sum_{\forall j \in Adj(i)} Z_{ij} = 2 \cdot Y_i, \qquad \forall i \in N \qquad (7.4)$$

where $Adj(i)$ is the set of nodes adjacent to node *i*, Y_i is the number of rings passing through node *i* and *N* is the set of nodes in the network graph. If the total number of ring modules is odd, then at least one additional ring module is required in a feasible ring cover. To illustrate this point consider the example in Figure 7.7.



(a) (b)

Figure 7.7. Ring parity condition: (a) working span loads, (b) possible ring cover.

Figure 7.7(a) shows the working load on the spans incident on a given node. For simplicity, only the spans incident on node 2 are shown Fig.7.7. If we assume that the ring capacity is 12 units, at least two rings are required to cover spans *(1,2)* and *(2,3)*, while three rings are required to cover span *(3,4)*. Because the total number of ring modules required is odd, however, a minimum of four rings is needed to cover the incident spans. This means that at least one excess ring module must be allocated to one of the spans. That is, the entire transmission capacity of one ring is completely unused on one of the incident spans. Figure 7.7(b) shows one possible ring cover. Note that three rings cover span *(1,2)*, although only two rings are required purely from a capacity point of view.

Another necessary condition for a feasible ring cover is that the number of ring modules on any one span incident on a node must not exceed the total number of modules on all other spans incident on the same node. This condition is called the *ring balance condition* and ensures that all rings incident on a given node are span-diverse. The ring balance condition can be expressed mathematically as:

$$Z_{ij} \le \sum_{\forall k \in Adj(i),\, k \ne j} Z_{ik}, \qquad \forall i \in N,\ \forall j \in Adj(i) \qquad (7.5)$$

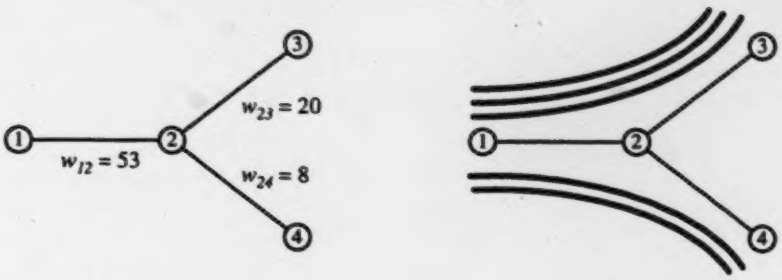
or, by rearranging terms as:

$$2 \cdot Z_{ij} \le \sum_{\forall k \in Adj(i)} Z_{ik}, \qquad \forall i \in N,\ \forall j \in Adj(i) \qquad (7.6)$$

To illustrate the ring balance condition, consider the example in Figure 7.8. The working load on the spans incident on node 2 are shown in Figure 7.8(a). Again, if we assume that the ring capacity is 12 units, the number of rings required on span *(1,2)* is five, while the total number of rings required on spans *(2,3)* and *(2,4)* is only three. Therefore, although the total number of ring modules required is even, at least two excess modules are required in a feasible ring cover. Otherwise, at least one ring would have to traverse span *(1,2)* twice, thereby violating the constraint on span diversity. One possible ring cover for this example is shown in Figure 7.8(b). Here, one excess ring module is allocated to spans *(2,3)* and *(2,4)*.

If we let $Z_{max}(i)$ denote the maximum number of ring modules required on any span incident on node *i*, then the minimum number of rings $Y_{min}(i)$ passing through node *i*, due to the ring balance condition alone, must satisfy the following inequality:

$$Y_{min}(i) \ge Z_{max}(i), \qquad \forall i \in N \qquad (7.7)$$



(a) (b)

Figure 7.8. Ring balance condition. (a) working span loads, (b) possible ring cover.

Combining Eq. (7.7) with Eq. (7.4), the minimum number of rings at each node that satisfies both conditioning arguments is

$$Y_{min}(i) = max\left(Z_{max}(i), \left\lceil \frac{\sum_{\forall j \in Adj(i)} \left\lceil \frac{w_{ij}}{m} \right\rceil}{2} \right\rceil\right), \qquad \forall i \in N \qquad (7.8)$$

and the number of excess ring modules E_i at node i is given by

$$E_i = 2 \cdot Y_{min}(i) - \sum_{\forall j \in Adj(i)} \left\lceil \frac{w_{ij}}{m} \right\rceil, \qquad \forall i \in N \qquad (7.9)$$

If the number of excess modules is less than two at all network nodes, obtaining a lower bound on the total transmission capacity is equivalent to solving an instance of the Chinese Postman Problem, as described in Section 6.3.3. Otherwise, a lower bound can be obtained by solving the following IP:

LBIP

Minimize:

$$\sum_{\forall ij \in S} Z_{ij} \qquad (7.10)$$

Subject to:

$$m \cdot Z_{ij} \geq w_{ij} \qquad \forall (ij) \in S \qquad (7.11)$$

$$\sum_{\forall j \in Adj(i)} Z_{ij} = 2 \cdot Y_i, \qquad \forall i \in N \qquad (7.12)$$

$$Z_{ij} \leq \sum_{\forall k \in Adj(i), k \neq j} Z_{ik}, \qquad \forall i \in N, \forall j \in Adj(i) \qquad (7.13)$$

$$Z_{ij} \geq 0 \text{, integer,} \qquad \forall i \in N, \forall j \in Adj(i) \qquad (7.14)$$

$$Y_i \geq 0\text{, integer,} \qquad \forall i \in N \qquad (7.15)$$

The objective (7.10) is to minimize the number of ring modules. Note that the objective can also be weighted by span distance, for example, to yield a lower bound on total fibre mileage. Constraint set (7.11) ensures that the aggregate capacity on any span equals or exceeds its working load. Constraint sets (7.12) and (7.13) ensure that the ring parity and ring balance conditions are satisfied, respectively. The AMPL model for this formulation is listed in Appendix E.

A lower bound on the number of ADMs in any network can also be found by taking the total demand originating/terminating at any node, dividing by the ADM add-drop capacity and rounding up to the nearest whole number. If there are no constraints on the ADM add-drop capacity, the lower bound on the number of ADMs A_{lb} is given by

$$A_{lb} = \sum_{i \in N} \left\lceil \left(\sum_{j \in N} d_{ij} \right) / (2 \cdot m) \right\rceil \qquad (7.16)$$

where d_{ij} is the demand between nodes i and j. For multi-technology designs, the minimum number of ADMs for each technology can be calculated by dividing the originating/terminating demand by the largest module size first. If the demand served by the last ADM consumes less than ¼ of its capacity, it is removed and replaced by the next size ADM. This takes into account the economy of scale effects in the cost model described in Section 7.3.2. That is, because a 4B48 ADM costs only twice as much as a 4B12 ADM, the break-even point for placing a 4B48 ADM occurs when the load equals or exceeds 24 STS-1s. Similarly, the break-even point between 4B48 and 4B192 ADMs occurs at 96 STS-1s of demand.

The total number of line regenerators can be found by dividing the span length l_{ij} by the regenerator spacing l_r, as follows:

$$\sum_{\forall ij \in S} \lceil l_{ij}/l_r \rceil \qquad (7.17)$$

The number of regenerators required at glassthroughs locations can be determined by subtracting the total number of ADMs from the total number of ring modules. Therefore, the total number of regenerators is given by

$$R = \sum_{\forall ij \in S} \lceil l_{ij}/l_r \rceil + \sum_{\forall ij \in S} Z_{ij} - A_{lb} \qquad (7.18)$$

Note that because the lower bounding procedure described above does not actually determine the number or placement of rings, the lower bound on the number of transitions must be zero. Using these conditioning arguments and by solving either IP, a lower bound on the total design cost can be calculated using Eq. (7.1).

This procedure is used in Chapter 12 to calculate a lower bound on total design cost for each test case. The actual usefulness of these bounds depends on how close they lie to the optimal solution. In addition, note that these lower bounds apply for the case where demands are routed in advance and, therefore, do not represent an absolute lower bound for the general case where routing is part of the decision space. To address this more general case, we reformulate the lower bounding IP to simultaneously optimize the routing of demands. This is done by generating a set of paths $P(k)$ for each demand k and adding decision variables for the flow over each path F_p. The formulation of this lower bounding procedure (with route optimization) is as follows:

LBRIP

Minimize:

$$\sum_{\forall ij \in S} Z_{ij} \tag{7.19}$$

Subject to:

$$m \cdot Z_{ij} \geq \sum_{p \in P(ij)} F_p \qquad \forall (ij) \in S \tag{7.20}$$

$$\sum_{p \in P(k)} F_p = d_k \qquad \forall k \in K \tag{7.21}$$

$$\sum_{\forall j \in Adj(i)} Z_{ij} = 2 \cdot Y_i, \qquad \forall i \in N \tag{7.22}$$

$$Z_{ij} \leq \sum_{\forall k \in Adj(i), k \neq j} Z_{ik}, \qquad \forall i \in N, \forall j \in Adj(i) \tag{7.23}$$

$$Z_{ij} \geq 0, \text{ integer}, \qquad \forall i \in N, \forall j \in Adj(i) \tag{7.24}$$

$$Y_i \geq 0, \text{ integer}, \qquad \forall i \in N \tag{7.25}$$

where $P(ij)$ is the subset of paths that intersect span *(ij)* and d_k is the quantity of demand k. Like the LBIP formulation, the objective (7.19) is to minimize the number of ring modules. Constraint set (7.20) ensures that the aggregate capacity equals or exceeds the total flow over any span. Constraint set (7.21) ensures that the total flow over all paths for a demand equals the its quantity. Constraint sets (7.22) and (7.23) ensure that the ring parity and ring balance conditions are satisfied, respectively. The AMPL model for this formulation is listed in Appendix E. This general formulation is also used in Chapter 12 to calculate lower bounds on the total design cost.

7.6.3 Statistical Inference

To assess the absolute performance of the design methods, we can also use statistical inference to derive point estimates of the optimal solution values for each test case, which – on the problems size *m*, are taken from a population whose minimum value is γ. It has been shown that if the minimum value in sample *i* is v_i, the distribution of v_i approaches a 3-parameter Weibull distribution as $m \to \infty$ [FiT28]. Strictly speaking, this result applies to continuous distributions only. Because the solution space for combinatorial optimization problems is typically quite large, however, it is reasonable to assume that the distribution of solution values is almost continuous. In addition, empirical results for the TSP show that there is no reason to reject the hypothesis that solutions obtained from repeated application of a heuristic are independent [Ree93]. In other words, each time a heuristic is used we implicitly sample a large number of possible solutions. Empirical studies suggest that this assumption can be applied to discrete combinatorial problems with a high degree of confidence [GoA79].

The basic idea here is that the distribution of feasible suboptimal solutions can only be one-sided with respect to the optimal solution and, therefore, must be truncated to zero at the optimal solution, as illustrated in Fig. 7.9.

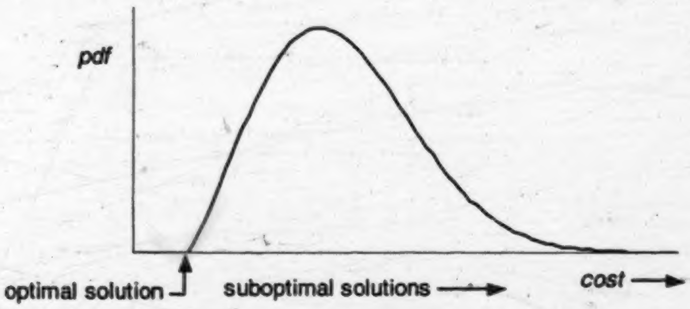


Figure 7.9. Distribution of suboptimal solutions relative to the optimal solution.

The cumulative distribution function for the 3-parameter Weibull distribution is given by the expression:

$$F(x) = P(X \le x) = 1 - \exp\left\{-\left(\frac{x-\gamma}{\alpha}\right)^{\beta}\right\}, \qquad \alpha > 0, \beta > 0, \gamma < x \quad (7.26)$$

where α is the scale parameter, β is the shape parameter and γ is the location parameter of the distribution. For each test case, we estimate these parameters using the procedure described in [Sta99]. This procedure involves rank-ordering the observations (i.e., solution values) and computing the *median rank* for each solution using the following expression:

$$F(t) = (j-0.3)/(n+0.4) \tag{7.27}$$

where $t = x-\gamma$, j denotes the solution order and n is the total number of solutions. The cumulative distribution function Eq. (7.26) is then converted to a linear equation $y = m \cdot x' + c$, where

$$y = \ln(\ln(1/(1-F(t)))) \tag{7.28}$$

$$x' = \ln(t) \tag{7.29}$$

$$m = \beta \tag{7.30}$$

$$c = \beta \cdot \ln(\alpha) \tag{7.31}$$

The Weibull scale α and shape β parameters can be estimated for a given value of the location parameter γ by fitting a regression line to the empirical data. Here, the quality of the linear fit is expressed by the correlation squared R^2. An estimate of the optimal solution value can be obtained by finding the value of the location parameter γ that maximizes R^2. For each test case, the optimal value of the location parameter is found using the "Goal Seek" function in Microsoft Excel™. This procedure is described in further detail in Appendix D.

A confidence interval $(1-\zeta)100\,\%$ for the optimal solution can also be obtained using the following expression [Ree93]:

$$w-\alpha/T<\gamma<w \tag{7.32}$$

where

$$T = \{-n/\ln(\zeta)\}^{1/\beta} \tag{7.33}$$

$$w = \min_i v_i \tag{7.34}$$

To improve the accuracy of the point estimate, intermediate solutions from the Tabu Search algorithm are also used in the sample. Point and interval estimates for the optimal solution for specific test cases are reported along with the other results in Chapter 12.

7.7 Summary

In this Chapter, we have described the test networks and modelling assumptions used to evaluate the design methods developed in the next four chapters. We have also established the metrics used to quantify the results and the methods for assessing the relative and absolute performance of each design method. These methods include a new lower bounding procedure and statistical techniques for obtaining estimates of the optimal solution.

8. Advances on the RingBuilder Approach: RingBuilder Interactive

8.1 Introduction

In this chapter we describe the software architecture and algorithmic details of RingBuilder Interactive, a complete network design tool developed to support current and on-going research on the multi-ring network design problem. The baseline heuristic algorithm implemented in RingBuilder Interactive arose in an immediately preceding M.Sc. thesis on RingBuilder by Slevinsky [Sle99], as described in Chapter 6. RingBuilder Interactive also embodies a number of new methods arising from this work.

The chapter begins with an overview of the software architecture and its main elements of RingBuilder Interactive. The heuristic algorithm used to synthesize designs is then described in Section 8.3 along with a detailed discussion of each of its main steps. In Section 8.4, we propose two new improvement heuristics to the baseline algorithm, followed by a brief summary in Section 8.5.

8.2 Software Architecture

This section provides an overview of the software architecture and data model for RingBuilder Interactive as of the end of this thesis work. This version we call RingBuilder Interactive or RBI 1.0. The development of this application represents the single largest breadth-type contribution of this thesis with over four person-years of effort. The application is written entirely in the Java™ programming language [Joy00]. Although Java is an interpreted language and is not as fast as compiled languages such as C/C++, it provides an object-oriented programming environment that is well-suited for rapid prototyping and also offers platform independence. These were two key requirements in the development of the application. Moreover it can now be compiled to native code for a variety of widely used target machines.

An architectural overview of RingBuilder Interactive is shown in Figure 8.1. The software architecture can be divided into five main components: the graphical user interface, the design algorithms, a report generator, file import/export utilities and the data model. The graphical user interface is used to visualize the input data and design results and control the design synthesis process. A screen capture of the graphical user interface is shown in Figure 8.2.

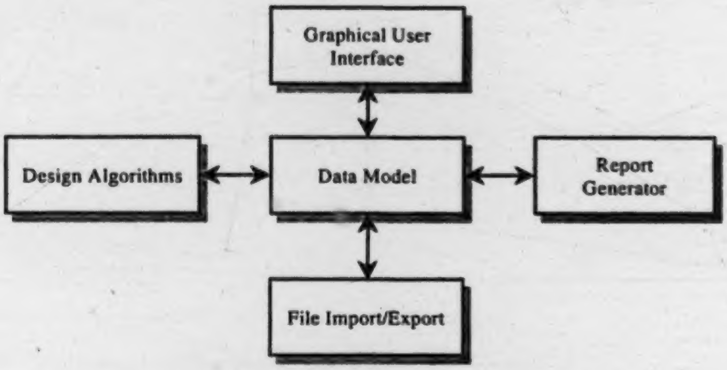

Graphical User Interface
Design Algorithms
Data Model
Report Generator
File Import/Export

Figure 8.1. RingBuilder Interactive software architecture.

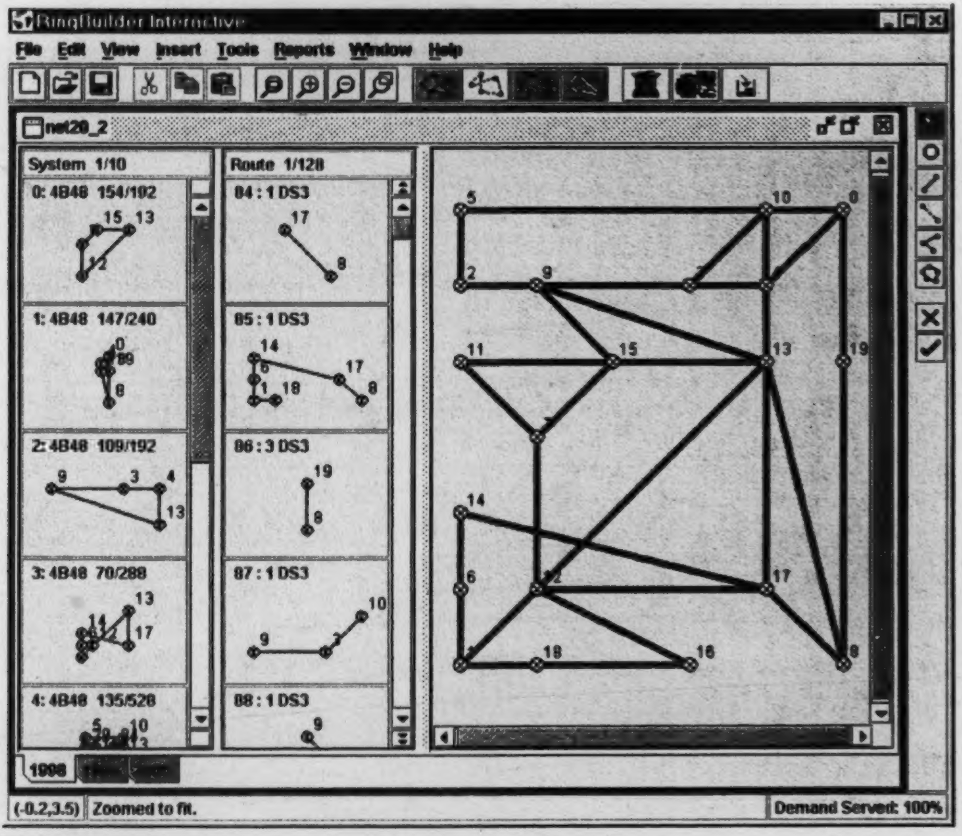

RingBuilder Interactive
File Edit View Insert Tools Reports Window Help
net20_2
System 1/10
0: 4B48 154/192
1: 4B48 147/240
2: 4B48 109/192
3: 4B48 70/288
4: 4B48 135/528
Route 1/128
84 : 1 DS3
85 : 1 DS3
86 : 3 DS3
87 : 1 DS3
88 : 1 DS3
1998
(-0.2,3.5)
Zoomed to fit.
Demand Served: 100%

Figure 8.2. RingBuilder Interactive screen capture.

The report generator provides a selection of standard reports including design summary information, detail routing information and various other useful reports. The file import/export utilities allow design input data and results to be imported and exported in several standard file formats. This provides backwards compatibility with previous versions of RingBuilder and Nortel's SONET Planner, a computer-aided ring design tool with no synthesis capabilities.

At the core of the tool is the data model used to represent the problem inputs (e.g., network topology, demand matrix and candidate ring types) and the ring network design output. A simplified diagram of the main data classes and their relationships is shown in Figure 8.3.

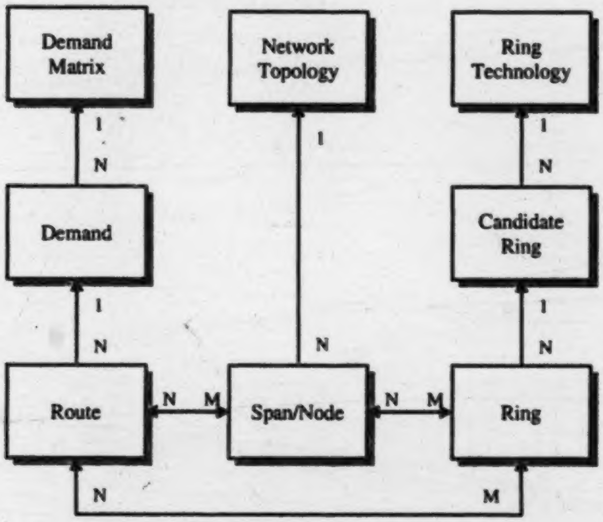


Figure 8.3. Diagram of main classes in RingBuilder Interactive.

The data model contains nine main classes. In Fig. 8.3, the lines with a single arrowhead indicate one-to-many relationships and the lines with two arrowheads indicate many-to-many relationships. For example, a one-to-many relationship exists between a Demand Matrix and a Demand because a demand matrix has many demands but each demand can belong to only one demand matrix. Similarly, a many-to-many relationship exists between a Route and a Ring because each Route may be carried by many rings and each ring may also carry many routes. For simplicity, the Span and Node classes are shown in a single box because they have similar relationships with other the classes.

The demand matrix contains a list of demands between origin-destination pairs. Each demand (or demand bundle to be precise) contains a reference to the origin and destination nodes, the data rate (e.g., DS3, STS-3c, etc.), the number of connections required at the specified data rate and a boolean flag for indicating whether the bundle may be split (or bifurcated) for routing purposes. Non-bifurcated routing is sometimes required in applications where the delay for all connections must be nominally the same (e.g., inverse multiplexing), or as a way of handling concatenated multi STS-1 SONET signals. The boolean flag along with the data rate and number of connections determine the demand's *minimum bundle size*. The minimum bundle size is the smallest amount of capacity by which the demand bundle can be divided. For example, the minimum bundle size of a demand of eight DS3s is either one DS3, if the route may be bifurcated, or eight DS3s (or 224 DS1 equivalents), if it cannot. Unlike all earlier versions of RingBuilder [Sle99], the current implementation supports both bifurcated routing for aggregate demands as well as multiple data rates (multiple demand unit capacities) in the same demand matrix. To support multiple data rates, all quantities are converted to DS1s within the application. For each demand, several routes may exist between the specified origin and destination nodes. A route contains a reference to its demand, a list of spans that it traverses and the flow (the number of DS3s, for example) that it carries. Note that the sum of flows over all routes for a given demand must equal the total demand. For example, if the demand between nodes A and Z is 17 DS3s, then the total flow over all routes between A and Z must also be 17 DS3s.

The network topology consists of a set of nodes and a set of fibre spans connecting the nodes. Each fibre span contains a reference to its two end nodes along with the distance (in kms) between the nodes. The current implementation supports parallel, physically separate spans between a pair of nodes. Each node contains a name, a pair of x-y coordinates and a reference to the cross-connect (or DCS), if any, located at the node. For simplicity, each node is equipped with at most one cross-connect, which is selected from a user-defined list of available cross-connect types. The cross-connect parameters include the total switching capacity (in DS1 equivalents), the fixed cost of the common equipment and the unit cost for each data rate supported by the cross-connect. The cross-connect and related classes are not shown in Figure 8.3. Note that if a node is not equipped with a cross-connect or the capacity of its cross-connect is exhausted, demands are not permitted to transit from one ring to another at the node.

Each network design problem also contains a user-defined list of ring technologies that may be included in the final design. Each ring technology is defined by its type (e.g., UPSR, BLSR) and line rate (e.g., OC-3, OC-12, OC-48, etc.). This may include 1+1 APS as a special type of two-node ring technology. Other ring technology parameters include the maximum number of ADMs in the ring, the maximum permitted ring circumference (in hops and kms) and a reference to the type of regenerator and ADM associated with the ring. The regenerator parameters include the maximum spacing (in kms) between adjacent regenerators and their fixed cost. The ADM parameters include the total add-drop capacity (in DS1 equivalents), the fixed cost of common equipment and the unit cost for each data rate supported by the ADM. Again for simplicity, the regenerator and ADM classes are also not shown in Figure 8.3. The combination of a ring technology and topological cycle represents a candidate ring from which actual rings may be instantiated. Each ring contains a reference to the ring type, line rate and the spans that it covers along with the set of routes served on each span of the ring. The portion (i.e., spans) of each route served by a ring is called a *route segment*.

In addition to the main classes described here, there are over 150 other classes used to encapsulate the behaviour of the graphical user interface, report generator, file/import export routines, and the design algorithms. Clearly, a complete description of all of these classes is beyond the scope of this thesis. A complete specification of all classes is contained in the RingBuilder Interactive API [TRL00a]. In the remainder of this chapter we focus on the algorithmic details of the heuristic algorithm used to generate multi-ring network designs. Further information about the other application features and options can be found in the TRLabs RingBuilder Interactive Users Guide also prepared by the author as part of this project [TRL00b].

8.3 Design Synthesis Algorithm

This section describes the basic heuristic algorithm used to synthesize network designs. The starting point for this algorithm was available on completion of the M.Sc. thesis by Slevinsky [Sle99]. Several new features and algorithmic enhancements are described in detail below.

8.3.1 Basic Overview

Like the original RingBuilder algorithm, the current algorithm consists of four main steps: (1) demand routing, (2) candidate ring generation, (3) candidate ring evaluation and (4) candidate ring selection. A flow chart outlining the basic procedure is shown in Figure 8.4.

The main inputs to planning problem are the demand matrix, network topology and the ring technologies under consideration. This data may be input via the graphical user interface or imported from plain text files. The first step in the basic procedure is to route the demands over the network topology. This is a critical step in the design process because it determines the spans within the network topology that must be covered the set of rings. Next, a set of candidate rings is generated from the network topology and the set of ring technologies. In practice, these first two steps may be performed independently.

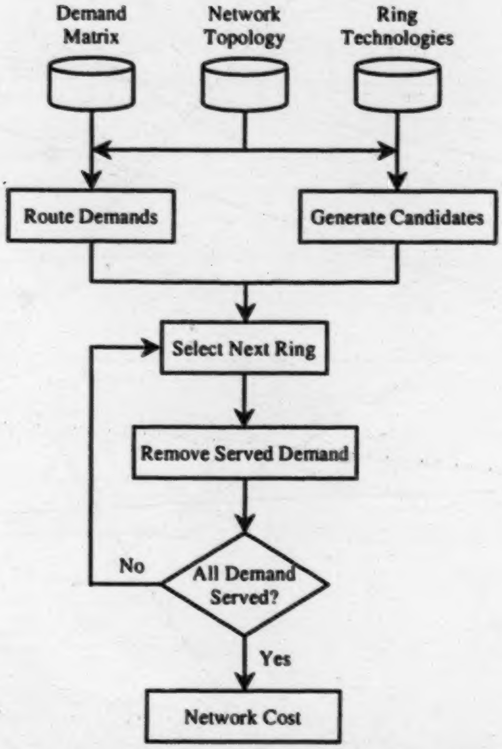

Demand Matrix
Network Topology
Ring Technologies
Route Demands
Generate Candidates
Select Next Ring
Remove Served Demand
No
All Demand Served?
Yes
Network Cost

Figure 8.4. Flow chart of the basic RingBuilder algorithm.

After the candidate rings have been generated and all demands have been routed, an iterative process is used to create a feasible solution one ring at a time. At each iteration of this process, the set of candidate rings is evaluated and the "best" candidate ring is selected and added to the design. An important subproblem of the evaluation process is determining the subset of routes that may be loaded onto each candidate ring. This step, known as *ring loading*, is described in detailed in Section 8.3.6. After the best candidate has been added to the design, the routes served by the ring are removed from the pool of (unserved) demand. The evaluation and selection process continues until all demands are served end-to-end. At the end of this process, the total design cost and other relevant design statistics are computed using the report generator.

In some cases, the design process can stall if none of the candidate rings is able to serve the remaining demand. This can occur when the candidate ring set does not cover the entire network graph and some nodes are isolated. Stalling can also occur when the DCS capacity at a node is exhausted and inter-ring routes cannot be connected to the remainder of the network.

8.3.2 Overview of Main Improvements

In previous versions of RingBuilder, ring loading is performed for the entire set of candidate rings at every iteration. This can be quite inefficient because the figure of merit for many of the candidate rings may not be affected by the previous ring selection. That is, if none of the route segments that were loaded onto a candidate ring is served by the selected ring, there is no need to re-evaluate the candidate ring because it will be faced with the same ring loading problem and its figure of merit will be unchanged. Candidate rings that serve no demand at all can also be removed from consideration for the remainder of the synthesis process. By selectively updating (i.e., re-loading) only those candidates affected by the previous selection and removing candidates that serve no demand, substantial reductions in runtime may be achieved. Normally this would involve storing the subset of route segments loaded onto each candidate ring and then finding the intersection between them. Because this requires a considerable amount of memory, a simpler approach is adopted in the current implementation. This simplification is based on the observation that two candidate rings can only contend for the same route segments if they share at least one common span. Thus, by re-evaluating only those candidates that intersect (i.e., share a common span with) the previously selected ring, we avoid the burden of having to store the route segments loaded onto all candidate rings. While slightly more rings may be evaluated than absolutely necessary, experience shows that this heuristic significantly reduces the amount of processing done at each iteration and greatly reduces total run-time.

The following subsections describe each of these design steps in detail.

8.3.3 Demand Routing

The purpose of the demand routing step is to assign demands to specified routes through the network topology. Like the previous versions of RingBuilder, the current procedure routes demands over the shortest path between the origin-destination demand pair using a min-heap implementation of Dijsktra's algorithm. The time complexity of this algorithm is $O(|N| \cdot \log|N|)$, where $|N|$ is the number of nodes in the network [MaG93], [Sha98]. In previous versions of RingBuilder, the demand routing procedure calls Dijkstra's algorithm once for each demand pair. This results in a worst-case time complexity of $O(|N|^3 \cdot \log|N|)$ for a full mesh demand pattern. In the current implementation, the demand routing procedure has been optimized to limit the number of calls to Dijkstra's algorithm. This relies on the fact that Dijkstra's algorithm actually computes the distance and (with some modification) the shortest path tree from the origin to all other nodes in the network. Therefore, the route for all demands containing the origin node can be enumerated from the shortest path tree. Using this approach, Dijkstra's algorithm is called $(|N| - 1)$ times at most and the worst-case time complexity of the revised demand routing procedure is $O(|N|^2 \cdot \log|N|)$.

Several routing options are also included in the current demand routing procedure. Demands may be routed based on either the number of hops in the path or geographical distance. In addition, the procedure may return either a single or multiple shortest paths between the origin and destination nodes. The latter option divides each demand bundle as evenly as possible between the k equally-shortest routes from the origin to the destination.

8.3.4 Candidate Generation

The purpose of the candidate generation step is to assemble a set of candidate rings from which the design is constructed. The candidate generation procedure begins by enumerating all cycles within the network topology. Several different algorithms have been proposed in the literature for enumerating all cycles in a network graph [MaD76]. Some of these work by generating all combinations of the fundamental cycle set and eliminating those combinations that do not form a cycle. In general, these methods are not very efficient because there are $2^{|S| - |N| + 1}$ such combinations, where $|S|$ is the number of spans in the network and $|N|$ is the number of nodes, and there is no apparent way to avoid generating all combinations.

Instead, we develop an efficient algorithm for enumerating all cycles in an undirected graph using a depth-first search. This cycle finding algorithm is particularly efficient because, unlike the algorithm used in previous versions of RingBuilder, each cycle is enumerated only once. The algorithm has a worst case time complexity of $O\{(|N| + |S|) \cdot (|C| + 1)\}$, where $|C|$ is the number of cycles in the graph. Because the number of cycles may be exponentially large, the algorithm also contains an option to limit the number of hops and/or circumference of the cycles. In most cases, this results in a dramatic reduction in runtime because large portions of the search tree need not be explored. This is a significant practical advantage of using a depth-first search for cycle enumeration. A detailed description of this algorithm is given in Appendix E.

For each combination of topological cycle and ring technology (i.e., the ring type and line rate), a *candidate ring* is formed. Thus, the total number of candidate rings is equal to the product of the number of cycles and the number of ring technologies under consideration. Each candidate ring represents a template (or blueprint) from which an actual ring may be instantiated. Note that the locations of ADMs and glassthroughs are not defined in the candidate ring. Instead these details are determined by the ring loading process described in the next section.

8.3.5 Candidate Ring Evaluation

In the candidate ring evaluation and selection step, the candidate rings are evaluated and the "best" one is selected for inclusion in the design. Defining the best candidate ring to add at each iteration is difficult because there are many factors that influence a ring's overall merit as part of a complete design. Furthermore, the greedy strategy of selecting the locally best candidate ring at each iteration does not guarantee a globally optimal solution.

In prior versions of RingBuilder, two metrics were proposed for evaluating candidate rings. The first [GSM95] uses a weighted sum of capacity utilization (balance) and the proportion of inter-ring transitions (capture) to assess the relative transport efficiency of ring candidates. One disadvantage of this approach is that the best balance-capture weighting depends heavily on the cost structure of the design problem. For example, in long-haul networks where distance is the dominant cost driver, the lowest cost designs are usually achieved with a high balance/capture weighting. Conversely, in metropolitan area networks a low balance/capture weighting is usually best because inter-ring transitions are an important cost element in the total design cost. This requires a rather intensive and time-consuming process of sweeping the weighting factor to find the best design. Even more importantly, the weighted balance-capture metric is not extensible to multi-technology designs because it does not account for differences in the cost of competing ring technologies and, in particular, economy-of-scale effects.

The second metric [Sle99] assesses the efficiency of a candidate ring by computing a "benefit-to-cost" ratio (also variously called a transport utility or specific cost measure). The benefit (in the numerator) is taken to be the sum of the flow-distance product of all routes that would be served by the candidate ring. This is a measure of the amount of the contribution that the candidate ring makes to completing the network design. The cost (in the denominator) of the candidate ring includes all costs associated with constructing the rings and serving the routes that it carries. The cost may be an arbitrarily detailed, highly realistic, assessment of the project cost of constructing the candidate ring including all specific circuit packs required at each node and so on. In general, this includes the cost of fibre, ADM common equipment, add/drop interfaces and inter-ring transitions. We refer, henceforth, to this benefit-to-cost ratio as the *transport efficiency* η_j of candidate ring j. This ratio can be expressed mathematically as

$$\eta_j = \frac{1}{c_j} \cdot \sum_{\forall i \in I(j)} l_{ij} \cdot F_i \qquad (8.1)$$

where c_j is the cost of the candidate ring j, F_i is the flow (in DS1 equivalents) carried on route i, l_{ij} is the distance (in either hops or physical distance) along route i between its entry and exit nodes on candidate ring j, and $I(j)$ is the subset of routes served by ring j. Note that the inverse of the transport efficiency is the cost per DS1-km (or DS1-hop) of demand served. Intuitively, this is an attractive measure of the utility of a candidate ring because it corresponds with the objective of the design problem, i.e. to minimize the total cost of the design subject to serving all demands from end-to-end. It also has a relevant economic interpretation. That is, the revenue from a transport service is usually a function of its data rate and the distance over which the service is provided. Therefore, the transport efficiency can be interpreted as the ratio of total revenue to total cost. Results show that single technology designs generated using this metric are typically as good or better and much more rapidly obtained than those generated using the balance-capture metric [Sle99]. For this reason, transport efficiency is adopted as a baseline in the current implementation of the basic algorithm.

We also introduce a new metric for ranking the candidate rings. This metric is similar to the transport efficiency metric given in Equation (8.1), except that the sum of the flow-distance product is raised to the power of an exponent, $a > 1$, as follows:

$$\zeta_j = \frac{1}{c_j} \cdot \left[\sum_{\forall i \in I(j)} l_{ij} \cdot F_i \right]^a \tag{8.2}$$

The effect of this metric is to give higher priority to candidate rings with larger flow-distance products. This is intended to compensate for the observed tendency of the previous metric to select several low capacity rings that are highly efficient in the local sense but not as efficient as a single, high-capacity ring in the long-run (due to economy-of-scale effects). We refer to this new metric as the *biased transport efficiency.*

8.3.6 Ring Loading for Candidate Evaluation

An important subproblem in the candidate ring evaluation and selection process is determining the subset of routes (or route segments) to load onto candidate rings. By a *route segment* we mean a subset of the spans in the route. In this section we develop two heuristic algorithms that find the subset of routes that maximizes the transport efficiency of each candidate ring within the given environment of currently unserved demands or demand segments. This is similar to the Ring Loading Problem defined in Section 4.4, except the current problem also involves finding the subset of nodes to equip with ADMs and the total flow over each route (or route segment) along with its entry and exit nodes. In addition, the problem environment includes several other practical details such as capacity constraints on ADMs and DCSs, demand splitting options and multiple data rates. These are details that were not addressed in Section 4.4, which touched mainly on the theory for strictly optimal loading. As a result, the IP formulations developed in that section are not directly applicable to the current problem. In fact, the objective for the current problem (i.e., maximizing the transport efficiency) is a non-linear function because the actual ring cost c_j (in the denominator) depends on the subset of nodes equipped with ADMs. Therefore, finding the strictly optimal solution using integer programming would involve solving each combination of ADM and glass-through nodes separately. Clearly this is not a practical approach because the number of combinations increases rapidly with ring size (i.e., number of nodes) and each IP subproblem may take several minutes to solve. Because thousands of candidate rings may need to be evaluated at each iteration of the synthesis process, heuristic algorithms were developed as they represented the only practical alternative. The two ring loading algorithms are now described in detail below.

8.3.6.1 Unbalanced Ring Loading Algorithm

The first ring loading algorithm is motivated by a simple, yet effective, heuristic for the one-dimensional Bin-Packing Problem. This problem involves packing a set of objects of different sizes into the least number of bins of fixed capacity. The basic idea of the heuristic, known as *first-fit decreasing* (FFD) algorithm, is to sort the objects in decreasing order of size and to pack them into the first bin with sufficient slack capacity [Mur92]. The current ring loading problem is similar to the Bin-Packing Problem except that only one copy of each candidate ring (i.e., bin) needs to be loaded per iteration. This is because the set of routes available for loading in each subsequent iteration depends on the ring selected in the current iteration. Therefore, it would be premature to load more than one copy of each candidate ring. For BLSR rings, the ring loading problem also includes more than one dimension because each span has its own capacity. Despite these differences, the basic FDD algorithm seems to be a reasonable approach. In the current context, the "size" of the objects (i.e., routes) is taken as the flow-distance product for each route. Thus, the basic loading rule is to load the "large, long" routes first and then fill the gaps with the "smaller, shorter" routes.

The main steps of the unbalanced ring loading procedure are listed in Figure 8.5. The first step in this procedure is to find the routes that intersect the ring. To facilitate this a hashtable is maintained that maps each span to a list of routes that intersect the span. The flow-distance product is computed by traversing each route to find the segments of the route that intersect the candidate ring and are not already carried by another ring. At the same time, the maximum flow through the ring is determined by checking the add-drop capacity on the ADMs and DCSs at the entry and exit nodes and the slack capacity on each span of the ring for BLSRs (or the slack capacity on the entire ring for UPSRs).

Step 1. Find the subset of routes that intersect the spans covered by the candidate ring and identify the associated route segments.

Step 2. Calculate the flow-distance product for each intersecting route and insert it into a maximum heap. Ties are broken in favour of the one with the longer route length.

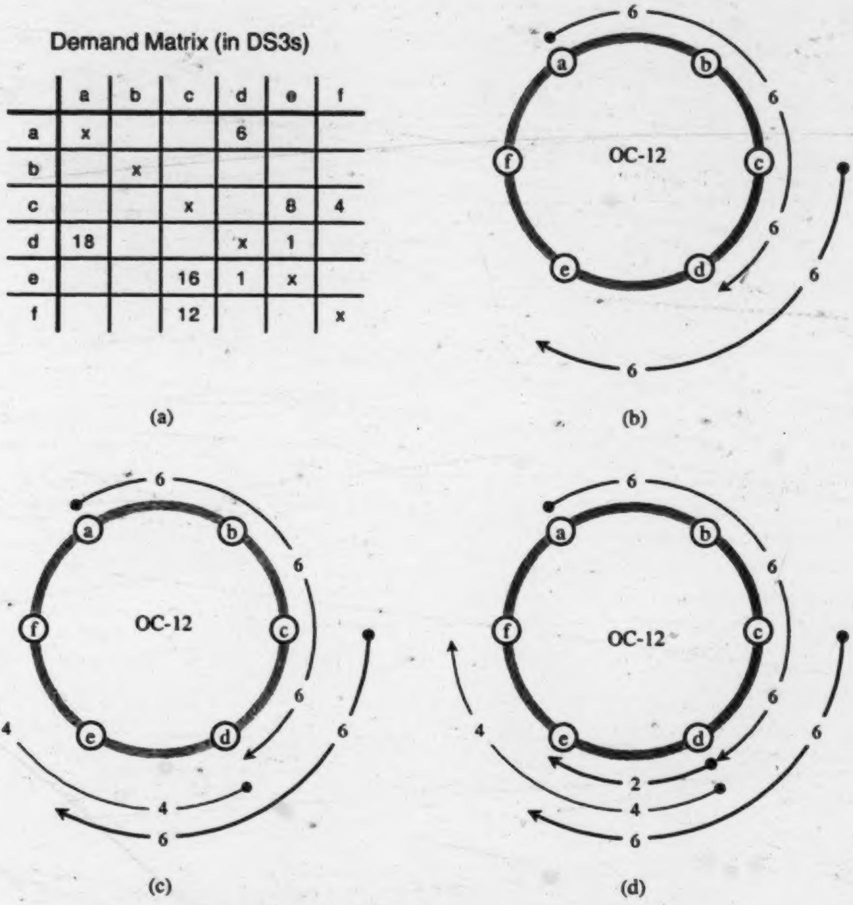
Step 3. Remove the next route from the top of the heap and attempt to load it onto the ring.

Step 4. Repeat step 3 until the all possible routes have been loaded (i.e., the heap is empty) or the capacity of the entire ring is exhausted.

Figure 8.5. Main steps in the unbalanced ring loading procedure.

Once the flow-distance product for a route is calculated, it is inserted into a maximum heap. The routes are then removed from the heap in decreasing order of flow-distance product and an attempt is made to load them onto the ring. This is done by traversing the segment of the route that intersects the ring and determining the maximum amount of flow through the ring. If the maximum flow equals or exceeds the route's flow, then the route segment is loaded onto the ring and the ring's working and slack capacities are adjusted accordingly. Otherwise, the route segment may either be partially loaded onto the ring or removed from consideration. If the maximum flow is greater than the route's minimum bundle size, then the original route is split in two routes, one that is loaded onto the ring and the other that is re-inserted into the heap after updating its flow-distance product. Otherwise, if the capacity at either of the end nodes or intervening spans is completely exhausted, alternate entry and exit nodes are identified and the entire route is re-inserted into the heap after updating its flow-distance product. Lastly, if the capacity along the entire route is completely exhausted, the route is removed from consideration. Note that this procedure does not attempt to balance the load around a BLSR by rerouting demands in the alternate direction around the ring, hence the name *unbalanced ring loading algorithm*.

At each iteration of this process, it takes $O(\log m)$ time to remove the next route from the heap and up to $O(|N|)$ time to traverse it, where m is the number of routes that intersect the ring and $|N|$ is the number of nodes in the network. In the worst case, all m routes may be evaluated up to $|N|$



Demand Matrix (in DS3s)

	a	b	c	d	e	f
a	x			6		
b		x				
c			x		8	4
d	18			x	1	
e			16	1	x	
f			12			x

(a) (b)

(c) (d)

Figure 8.6. Example of the unbalanced ring loading procedure.

times, so the computational complexity of the algorithm is $O(m \cdot |N| \cdot (|N| + \log m))$.

To illustrate the ring loading procedure, consider the example in Figure 8.6. Here four routes intersect a 4-fibre OC-12 BLSR (4B12) with six nodes. The demand matrix is shown in Figure 8.6(a) and all demands are assumed to be routed in the clock-wise direction. For convenience, the quantity of demand (in DS3s) is given in the upper right half of the matrix and the flow-distance product is given in the lower left half. Assuming that there are no ADM or DCS capacity constraints at any of the nodes on the ring and that all routes may be split, the loading algorithm proceeds as follows. First, the route with the highest flow-distance product, route (a,d), is selected and all 6 DS3s are loaded onto the ring. Then route (c,e) is selected for loading. Because the capacity remaining on span (c,d) is only 6 DS3s, route (c,e) is split into two routes, one with a flow equal to

6 DS3s and the other with a flow equal to 2 DS3s. The former is then loaded onto the ring and the latter is re-inserted into the heap. Because the latter route cannot be loaded onto span (c,d) due to capacity exhaustion, span (c,d) is removed from the route segment and its flow-distance product is reduced to 2 DS3-hops. The routes loaded onto the ring up to this point are shown in Figure 8.6(b). Next, route (c,f) with a flow-distance product of 12 DS3-hops is removed from the heap. Because it cannot be loaded on span (c,d), its flow-distance product is reduced to 8 DS3-hops and is re-inserted into the heap. Because route (c,f) still has the highest flow-distance product of the remaining routes, it is once again removed from the heap and then loaded onto spans (d-e) and (e-f) on the ring, as shown in Figure 8.6(c). At this point, only two routes remain in the heap, route (d,e) with a flow-distance product of 1 DS3-hop and the remaining portion of route (c,e) also with a flow-distance product of 2 DS3-hops. The latter route is removed from the heap and loaded onto ring, as shown in Figure 8.6(d). Because the capacity on span (d,e) is now exhausted, the remaining route (d,e) is removed from consideration and the ring loading procedure terminates.

Using this loading procedure, the total flow-distance product served by the ring is 40 DS3-hops. A total of five ADMs, 36 add-drop ports and 6 inter-ring transitions are required. Based on the cost model in Chapter 7, the total cost of the ring is $6.2X$ (excluding cost of fibre and regenerators) and its transport efficiency is

$$\eta = 40/6.2X = \frac{6.45}{X} \tag{8.3}$$

Note that the locations of the ADMs are determined by the subset of routes that are loaded onto the ring. That is, for every route that is loaded an ADM must exist (or be added) at the entry and exit nodes. For the example shown in Figure 8.6(d), ADMs are required at nodes a, c, d, e and f. Node b, on the other hand, does not require an ADM because none of the routes that were loaded enter or exit the ring at that node. While this eliminates the combinatorial problem of considering every combination of two or more ADMs on each candidate ring, there is nothing to suggest that the present procedure chooses the optimal set of ADMs. For example, the current procedure may add an ADM to a ring that adds or drops only a single DS3. This may be particularly inefficient if another ring is placed in a subsequent iteration that would be better suited to handling the load.

There are two important differences between the current ring loading procedure and those implemented in previous versions of RingBuilder. In previous versions of RingBuilder, routes are either treated as a single bundle or divided by the minimum bundle size and handled as individual routes. Although handling each route as a single bundle reduces the number of routes to consider during the ring loading process, it can be grossly inefficient if the demand is large relative to the ring capacity. Conversely, dividing each route into individual routes improves capacity efficiency but at the cost of increased computation time. Therefore, the strategy adopted here is to keep routes bundled as long as possible and split them only as required. This reduces the number of routes in total, yet retains the capacity efficiency of handling each unit of demand separately. The current ring loading algorithm also re-evaluates the capacity-distance product of each route prior to loading. This reflects the fact that the preferred order in which routes are loaded may be affected by prior loading decisions. Thus, the hypothesis is that dynamically updating the order in which the routes are loaded should improve the quality of loading decisions. In previous implementations, the order in which routes are loaded is not dynamically updated.

One limitation of the basic ring loading procedure for BLSR rings is that it does not consider routing demands in the alternate direction around the ring (i.e., load balancing). Preliminary results show that the lack of loading balancing can results in up to half the ring capacity remaining unused. In addition, routes that intersect the ring at two or more nodes but do not share the same sequence of spans are not loaded onto the ring. Generally, this results in more rings being placed than absolutely necessary. Although these deficiencies are mitigated to a large degree by the demand packing procedure described in Section 8.4.1, we also develop a *balanced* ring loading procedure in the next section that more directly addresses these issues.

8.3.6.2 Balanced Ring Loading Algorithm

The balanced ring loading procedure described in this section was developed primarily for BLSR rings but the basic procedure can also be adapted to UPSR rings. At a high level, the balanced ring loading procedure is almost identical to the basic ringloading procedure, as described in Fig. 8.5. The main difference lies in the definition of an intersecting route. Here, we define an intersecting route as any route that has at least two nodes in common with the ring. This allows routes that straddle the ring to also be considered during the ring loading process.

After the intersecting routes have been identified, each one is traversed to find the first and last nodes where the route may enter and exit the ring. This involves checking the add-drop capacity on the ADMs and DCSs at intersecting nodes along the route. The entry and exit nodes are also chosen so that they do not fall within a portion of the route that is already carried by another ring. Although technically feasible, this would involve verifying that the route could be dropped from the other ring at the same node and would greatly complicate the ring loading procedure. The entry and exit nodes are, however, allowed to completely surround an already served portion (or portions) of the route. Technically, this represents a duplication of effort because capacity is already allocated for the route in the previous ring. However, it is generally desirable because it reduces the number of inter-ring transitions.

Once the entry and exit nodes have been determined, a separate *route segment* is created for each direction around the ring between the entry and exit nodes. The route segments contains a reference to the original route and the spans on the ring between the entry and exit nodes. Note that the sequence of spans in at least one of the route segments will be different from those in the original route. In effect, this represents an alternate path for the route between the entry and exit node.

To determine the priority order in which route segments are loaded onto the ring, we take the flow-distance product of the route's original path between the entry and exit nodes (less any portions that have already been served) and divide it by the cost of the resources consumed along the ring. The priority p_{ij} of loading route segment *i* on ring *j* can be expressed mathematically as:

$$p_{ij} = (l_{ij} \cdot F_i)/c_{ij} \tag{8.4}$$

where l_{ij} is the distance along route segment's *i* original path (less any portions that have already been served) between the entry and exit nodes on ring *j*, F_i is the flow on route segment *i* and c_{ij} is the cost of the resources consumed on ring *j* by route segment *i*. Ties are broken by giving priority to route segments with the largest flow. The cost c_{ij} includes the cost of the transmission capacity, inter-ring transitions and a fraction of the fixed cost of the ADMs, if required at the entry and exit nodes. The cost of the transmission capacity is taken as the product of the flow-distance product along the ring multiplied by an average cost per unit flow-distance. This recognizes that the resources (i.e., capacity) consumed in both directions around the ring may be significantly different and gives preference to the shorter of the two directions. The inter-ring transition cost is equal to the number of inter-ring transitions multiplied by the cost of the add-drop interface at the corresponding DCS(s). If an ADM is not present at the entry or exit node, a specified fraction of the ADM fixed cost is also added to the cost estimate. We refer to this fraction as the *ADM discount factor*. This has the effect that route segments that require additional ADMs will have a lower priority than those that do not, all other things being equal. Note also that route segments do not get credit for those portions between the entry and exit nodes that are already served. Therefore, routes that are not already served between the entry and exit nodes will also have a higher priority, all other things being equal.

Once the priority of the route segments is calculated, they are inserted into a maximum heap. The routes are then removed from the heap in decreasing order of priority and an attempt is made to load them onto the ring. At this time, the maximum flow along the ring is determined by checking the slack capacity at the entry and exit nodes and on each span of the BLSR. If the maximum flow along the ring is greater than the route's minimum bundle size, the spans in the route between the entry and exit node are replaced by those in the route segment and the flow is set to the maximum flow or the route's flow, whichever is less. The route is then allocated to the ring and the capacity along the ring and at the entry and exit nodes is updated accordingly. If all of the flow is satisfied, the corresponding route segment in the alternate direction is discarded. Otherwise, it is retained and the flow is set to the remaining unserved flow. Each time an ADM is placed during the loading procedure, the heap is reordered to reflect possible changes in the relative priority of the remaining route segments. This loading procedure continues until the ring capacity is exhausted or the heap is empty. It is easy to show that the computational complexity of this loading algorithm is also $O(m \cdot |N| \cdot (|N| + \log m))$. In the current algorithm, however, the number of intersecting routes is typically much larger than in the basic ring loading algorithm.

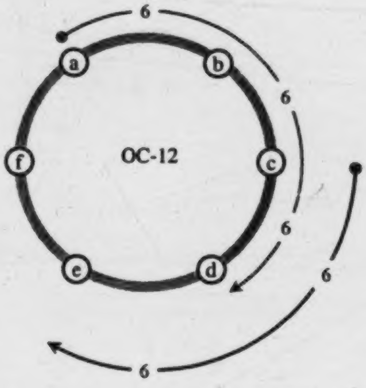
Figure 8.7 illustrates the balanced ring loading procedure for the previous example given in Figure 8.6. For simplicity, we assume that all route segments originate and terminate on the ring and that the original routes traverse the spans on the ring (i.e., they do not straddle the ring). The cost of adding a new ADM (i.e., ADM discount factor) is also set to 1/10th of its fixed cost. Based on these assumptions, the loading priority, p_{ij}, of the route segments for each iteration of the loading procedure is shown in Figure 8.7(a). Assuming that there are no ADM or DCS capacity constraints at any of the nodes on the ring and that all routes may be split, the loading algorithm proceeds as follows:

In the first iteration, route segments (a,d) and (d,a) have the highest priority. Therefore, route segment (a,d) is selected arbitrarily and all 6 DS3s are loaded onto the ring. Because this completely satisfies the flow on route (a,d), the alternate route segment (d,a) is set to zero and removed from the heap. Note that to load this route segment onto the ring, ADMs are required at nodes a and d. The placement of these ADMs and, in particular, the one at node d, reduces the number of ADMs required for route segments (d,e) and (e,d) and increases their priority in the next iteration. Because the other route segments do not share a node with route segment (a,d), their priorities remain the same. Next, route segment (c,e) with the second highest priority is selected for loading in iteration 2. Because 6 DS3s of capacity are already allocated on span (c,d), the flow on route segment (c,e) is reduced from 8 DS3s to 6 DS3s and the flow on the alternate route segment (e,c) is set to 2 DS3s. Because this adds another two ADMs to the ring at nodes c and e, the priority of the remaining route segments are adjusted accordingly. The routes loaded onto the ring up to this point are shown in Figure 8.7(b). In the third iteration, route segment (d,e) with a flow of 1 DS3 is added to the ring and its corresponding route segment (e,d) is removed from the heap. At this point, route segments (c,f) and (f,c) have the highest priority. Because the capacity is exhausted on span (c,d), however, route segment (c,f) cannot be loaded and is therefore discarded. Instead, route segment (f,c) with a flow of 4 DS3s is loaded onto the ring, as shown in Figure 8.7(c). Because 2 DS3s of capacity are still available from node e to node c, route segment (e,c) is loaded in its entirety in the last iteration. The final loading pattern is shown in Figure 8.7(d).

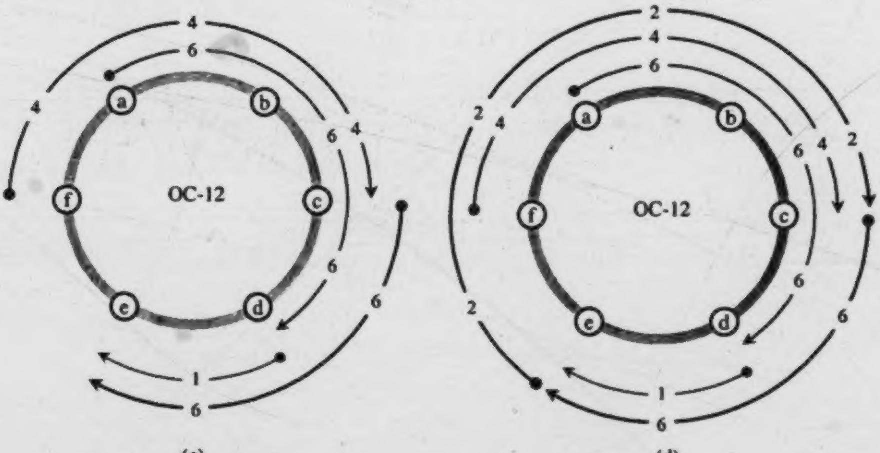
Route Segment Loading Priority, p_{ij}

Route Segment	Iteration				
	1	2	3	4	5
(a,d)	10.6				
(d,a)	10.6				
(c,e)	10.4	10.4			
(e,c)	5.6	5.6	6.0	6.0	6.0
(c,f)	10.0	10.0	10.9	10.9	
(f,c)	10.0	10.0	10.9	10.9	
(d,e)	3.5	5.5	12.0		
(e,d)	1.6	1.9	2.4		

(a)



(b)



(c)

(d)

Figure 8.7. Example of the balanced ring loading procedure.

Using the balanced ring loading procedure, the total flow-distance product served by the ring is 47 DS3-hops. A total of five ADMs and 38 add-drop ports are required. Unlike the unbalanced ring loading procedure, all of the demand is served and there are no inter-ring transitions. Based on the cost model in Chapter 7, the total cost of the ring is 5.95X (excluding cost of fibre and regenerators) and its transport efficiency is

$$\eta = 47/5.95X = 7.9/X \tag{8.5}$$

The transport efficiency in this case is 22% higher than that attained by the unbalanced loading procedure.

8.3.7 Candidate Ring Selection

We also developed two approaches for choosing the candidate ring to add at each iteration. The first approach simply selects the candidate ring with the highest (biased) transport efficiency. We refer to this approach as the *greedy selection* method. One disadvantage of this approach is that only one design can be generated for a given design problem. This limits the exploration of the solution space to a single point. Furthermore, selecting the best candidate at each iteration of the design process does not guarantee a globally optimal solution.

As an alternative, we propose a *probabilistic selection* method to randomize the choice of candidate ring at each iteration. To do this we sort the candidate rings in decreasing order of transport efficiency and assign a probability distribution P_k for selecting the k^{th} ranked candidate ring. To allow the probability distribution to be calculated quickly, it is defined over a limited number of top ranked (or *elite*) candidate rings. This simplification is based on the high probability of choosing one of the elite candidates. The probability distribution is also biased to account for differences in the transport efficiency of the elite candidates. For example, when the transport efficiencies of the top-ranked candidates are closely clustered, the probability of selecting any given candidate is roughly the same. Conversely, when there is a wide spread in the efficiency of the elite candidates, the probability distribution is biased to favour those candidates with higher efficiencies. The rationale for this approach is that the probability of selecting a candidate should be tied to its relative efficiency. The function adopted for this purpose is

$$P_k = \frac{(\eta_k/\eta_1)^x}{\sum_{i=1}^{n} (\eta_i/\eta_1)^x} \tag{8.6}$$

where η_i is the transport utility of the i^{th} ring, n is the number of elite candidates and $x \geq 0$ is an exponent used to control the "greediness" of the selection. The effect of the exponent x on the probability distribution is illustrated in Fig. 8.8. In this figure, the transport efficiency of the top ranked rings decreases linearly as a function of *k*. When the exponent $x = 0$ the probability of selecting any of the top ranked rings is the same. As the exponent increases, the probability distribution becomes skewed to the higher ranked rings.

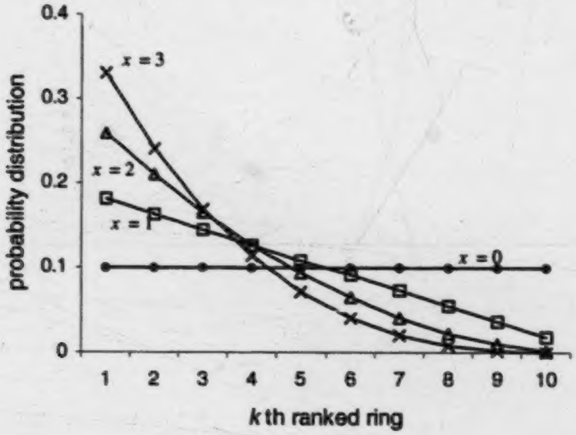


Figure 8.8. Effect of exponent *x* on the selection probability distribution.

Note that when the transport efficiency is the same for all *n* candidate rings, the probability distribution is uniform regardless of the exponent value, as desired.

Using this selection method, a number of solutions can be generated for the same design problem. Randomizing the selection also helps to combat "noise" inherent to the candidate evaluation process. Initial results using this selection method show that as the design approaches completion it is usually advantageous to revert to the greedy selection method. Therefore, a second parameter was added to determine the point at which to switch from the probabilistic to greedy selection method based on the percentage demand served. This selection method is an integral part of the Dithered Sequencing design approach described in Section 8.4.2.

8.4 Improvement Heuristics

This section proposes two new heuristics for improving the quality of solutions generated by the basic algorithm. The first is a demand packing algorithm that attempts to route additional demand within the existing rings at intermediate stages in the design process. This algorithm is similar in purpose to one developed by Owen [Owe96] but differs significantly in the details. The second improvement heuristic, called *dithered sequencing*, explores multiple design sequences to diversify the basic search procedure. The demand packing and dithered sequencing heuristic algorithms are described in detail below.

8.4.1 Demand Packing

The purpose of the demand packing algorithm is to route (or *pack*) unserved demands within the slack (or unused) capacity that arises as rings are added to a design. This slack occurs when the rings added to the design are not completely filled due to static demand routing and/or ring loading inefficiencies. Because the amount of slack within a partially complete design can be quite significant, demand packing offers the prospect of serving additional demand at little or no extra cost. It may also eliminate the need for additional rings as a design nears completion and only a few demands remain unserved.

The demand packing problem is a special case of the *multicommodity maximum flow problem* [Hu82] in which the flow variables are integer. The objective is to route the maximum amount of demand through the network without exceeding the ring capacities. Traditional methods for solving multi-commodity flow problems are based on linear-programming models, which can be solved in polynomial time using ellipsoid or interior point methods [GOP98]. Because the coefficient matrix for the multi-commodity maximum flow problem is not totally unimodular, however, it may not have an integer optimum solution even if the data are all integral [Mur92a]. Furthermore, because the demand packing algorithm may be called frequently by the main procedure, the run-time of the algorithm is a key consideration. For these reasons, a heuristic algorithm based on *k*-shortest path routing of all demands was used for demand packing. Prior work by Dunn et al. [DGM94] on the two-terminal multicommodity maximum flow problem for span restorable mesh networks found this approach to be quite successful. The main steps of the present algorithm is listed in Figure 8.9.

In the first step of the demand packing algorithm, an undirected graph is constructed that represents the connectivity and capacity constraints of the existing network facilities (i.e., rings and DCSs). This is done by explicitly modeling all the network elements and the connections between them. We begin by adding the existing ring set to the graph. For each ring, we add to the graph a set of vertices that correspond to the ADMs and glassthroughs in the ring and a set of edges that correspond to the spans between them. Associated with each edge is a capacity and a cost. The capacity is determined by the utilization of the underlying ring. If it is a BLSR, the edge capacity is equal to the slack on the corresponding span. If it is a UPSR, the capacity is equal to the slack capacity for the entire ring. A cost per DS1 is also computed for each edge based on the fixed cost and total capacity of the ring. This cost represents the cost of consuming ring capacity. These vertices and edges represent the intra-ring connectivity and capacity constraints.

Step 1. Construct an undirected graph that represents the intra-ring and inter-ring connectivity.

Step 2. Sort the demand bundles in decreasing order of the amount of demand (or flow) that remains uncarried.

Step 3. (a) Select the demand bundle with the highest outstanding flow and find the shortest (i.e., least-cost or min-hop) path from the origin to the destination node whose maximum flow exceeds the minimum bundle size.

(b) Route as much of the demand as possible over the shortest path.

(c) Repeat steps 3(a) and 3(b) until either all of the demand is served or there is no path that meets the minimum bundle size requirement.

(d) If all remaining demand is served, remove the original routes that were not served from end-to-end. Otherwise, distribute the remaining demand as evenly as possible amongst the original incomplete routes.

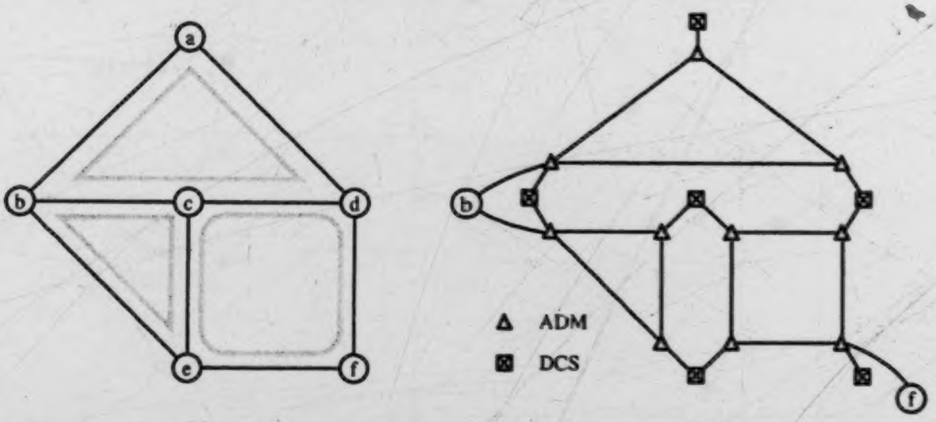
Step 4. Repeat Step 3 for all demand bundles.

Figure 8.9. Simplified the demand packing algorithm.

Next, for each DCS in the network another vertex is added to the graph. These vertices are then connected to the vertices representing co-located ADMs and glassthroughs (i.e., ADMs and glassthrough that share the same node as the DCS). The edges that connect these network elements also have a capacity and cost associated with them. The capacity of an ADM-DCS edge is equal to the slack add-drop capacity on either the DCS or its adjacent ADM, whichever is less. The cost of the ADM-DCS edges is equal to the sum of the add/drop interface costs on the ADM and DCS. Note that this cost depends on the data rate of the demand being served. The capacity of glassthrough-DCS edges is equal to the capacity of an ADM for the given ring type or the slack add/drop capacity of the DCS, whichever is less. These edges are included to allow glassthroughs to be replaced with ADMs when economically viable. The cost of a glassthrough-DCS edge is the same as an ADM-DCS edge, except there is an additional charge for placing the ADM. As in the balanced ring loading algorithm, this additional charge is a specified fraction of the fixed cost of an ADM. Therefore, if the least-cost path traverses a glassthrough-DCS edge, the glassthrough will be replaced by an ADM, provided the maximum number of ADMs for the ring is not exceeded. Otherwise, the cost of the glassthrough-DCS edge is set to infinity. The working assumption here is that all inter-ring demands pass through a DCS. Thus, the vertices and edges added in this step represent the inter-ring connectivity and associated capacity constraints.

The last step involves adding a vertex to the graph for each network node. These vertices represent the sources/sinks of demand at the origin and destination nodes for each demand pair. To prevent inter-ring transitions from occurring through these vertices, they remain unconnected from the rest of the graph until the shortest path between an O-D pair is requested. At that point, the vertices for the origin and destination nodes are connected to their co-located ADMs and glassthroughs. The capacity on each of these source/sink-ADM edges is equal to the slack on the adjacent ADM and the cost is equal to the cost of an add/drop interface on the ADM. This is based on the assumption that the sources/sinks of demand are connected directly to an ADM rather than being connected via a DCS. Otherwise, the vertices representing these nodes could have been connected directly to the co-located DCSs. Once the shortest path(s) for an O-D pair have been found, these edges are removed from the graph. This approach also allows demands to be added/dropped at nodes (locations) that are not equipped with a DCS.

Figure 8.10 provides an example of a ring set and its corresponding ring connectivity graph. The original network topology and ring set (shaded) is shown in Figure 8.10(a) and the corresponding ring connectivity graph for demand pair (b,f) is shown in Figure 8.10(b). Note that the other network nodes are omitted from the ring connectivity graph in Figure 8.10(b) mainly for illustrative purposes but also because they are not connected to the rest of the graph.



(a) (b)

Figure 8.10. An example of a ring connectivity graph used in demand packing of existing rings.

The second step in the demand packing algorithm is to sort the demand bundles in decreasing order of the unserved demand. By an unserved demand, we mean the amount of demand (in DS1 equivalents) that is not currently carried by a ring(s) from end-to-end. The rationale for sorting the demands in this order is that it should improve packing efficiency if the larger demands are packed before smaller ones.

Next, for each O-D demand pair, a variant of Dijkstra's algorithm is used to find the shortest path between the end nodes. The path length may be measured in terms of cost or the number of hops. The cost of a path is equal to the sum of the cost of all edges traversed along the path. This cost model is based on the principle that there is an *opportunity cost* associated with consuming network facilities. That is, by allocating capacity to a particular demand pair, we forego the opportunity of using it for other purposes (e.g., leasing).

If the slack capacity on a given edge is less than the minimum bundle size of the demand being routed, the cost of the edge is set to infinity. If a shortest path is found that satisfies the minimum bundle size requirement, as much demand as possible is routed over this path and the slack capacity is adjusted accordingly. If some of the demand remains unserved, the shortest path between the O-D pair is re-computed using the updated slack values. This process continues until either all of the demand for the given O-D pair is served or no further paths exist. If all of the demand for the O-D pair is served, the original (incomplete) routes for this demand pair are permanently removed from the route table, having been completely replaced by the new routes. This frees up additional capacity for routing subsequent demands, either in the current or future demand packing intervals. If none of the demand is served, the original routes for this OD-pair are left intact. Otherwise, if some fraction of the demand is served, the flows on the original routes are decreased by the amount served. This is done by reducing the flow on the original routes in proportion to the original distribution so that the total reduction equals the total flow on the new routes that were created via the packing process. Because the amount on each route must be a whole number, the proportion of demand allocated to each route is adjusted slightly to account for rounding errors. The method used guarantees that the total demand is reduced by the correct amount and that the reduction in demand is spread as evenly as possible amongst the original routes.

An alternative to the above approach, is to remove all of the incomplete routes prior to starting the demand packing process. This makes additional capacity available for demand packing and should, in theory, make it possible to pack more demand in total. But it also makes it necessary to reconstruct the original routes if only a fraction (or none) of a demand pair can be served. Due to the complexities involved in doing this, we opted for the simpler approach described above. To take full advantage of the capacity released after each demand is packed, however, we place the demand packing procedure in a loop that continues to iterate until no further demand can be packed. This increases the amount of demand served at each packing interval, while avoiding the complexities of reconstructing routes.

To automate the demand packing process, an option was added to the basic RingBuilder algorithm to initiate demand packing at specified intervals during the design process. The demand packing interval may be based on either the number of iterations (i.e., rings placed) or the percentage of demand served.

It should also be noted that the demand packing algorithm described above has several other possible uses in the context of ring network planning. For example, it can be used in a multi-period planning problem to utilize the slack in existing rings (placed in prior periods) before adding new rings to accommodate demands in the current and subsequent periods. It can also be used to determine a routing pattern in situations where only a set of rings is specified (e.g., in manual ring planning). Lastly, the demand packing algorithm is used in the Tabu Search algorithm described in Chapter 11.

8.4.2 Dithered Sequencing

The purpose of the dithered sequencing heuristic is to diversify the search procedure of the basic RingBuilder algorithm by simultaneously exploring several design sequences (or trajectories) through the design space. A *design sequence* is simply the sequence of ring choices made in constructing the solution. By randomizing (i.e., dithering) the ring selected at each iteration of the design process, we can construct several solutions to the same design problem. This overcomes an inherent limitation of the basic greedy algorithm, which generates only a single solution for each design problem. To randomize the choice of ring at each iteration we use the probabilistic selection method described in Section 8.3.7. For each design problem, we explore several design sequences and compare their progress at regular intervals during the design process. At the end of each interval, the least promising of the design sequences are abandoned and several new design sequences are spawned from those that survive the "pruning" process. This process of design sequence pruning and branching continues until each of the remaining design sequences serves all of the offered demand. The basic procedure is illustrated graphically in Figure 8.11.

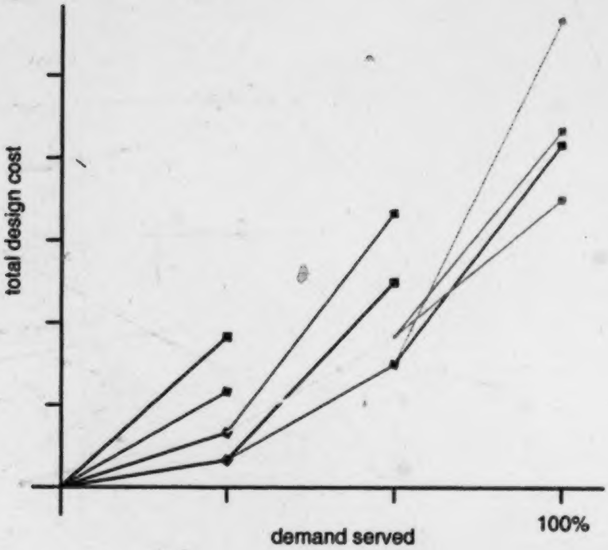


Figure 8.11. Illustration of the basic Dithered Sequencing meta-heuristic.

In the current implementation, the points at which pruning and branching occur are demarcated by the percentage of demand served. That is, each design sequence continues adding rings until the percentage of demand served for the next interval is reached. These *branching points* occur at regular intervals during the design process. For example, if the design is separated into four stages, pruning and branching occur when the percentage of demand served reaches the 25%, 50% and 75% levels. The number of branching points, n, the number of elite sequences per branching point, r, and the number of branches per elite solution, k, are user-defined parameters. In Figure 8.11, for example, $n = 3$, $r = 2$ and $k = 2$. Note that at any stage of the design process, the number of design sequences is $r \cdot k$.

Because there is a step-wise increase in the amount of demand served with each ring added to the design sequence, the percentage of demand actually served by a design sequence usually exceeds the desired amount by some margin at the branching point. Therefore, to compare the relative efficiency of the design sequences, the cost of each design sequence is divided by the percentage of demand that it serves. For example, if a design sequence that serves 25% of the demand has a total cost of 100X, its efficiency is 400X. This means that given the current rate of progress the total cost of the design would be 400X. The effect of different parameter settings on solution quality is considered in the experimental results.

8.5 Summary

In this chapter, we described the software architecture and algorithmic details of RingBuilder Interactive. We also introduced two new heuristics designed to improve the performance of the baseline approach. In the next chapter, we present the study method used to compare the performance of these heuristics along with the detailed results.

9. Results of RingBuilder Improvements

9.1 Introduction

The previous chapter described the baseline RingBuilder algorithm and proposed several improvement heuristics. In this chapter, we compare the performance of these improvement heuristics (relative to the baseline algorithm) for the test cases outlined in Chapter 7. This chapter begins by describing the study method used to characterize the performance. The results are presented in Section 9.3, followed by a summary and conclusion in Section 9.4.

9.2 Comparative Study Method

This section describes the experimental methods used to evaluate the performance of each improvement heuristic relative to the baseline algorithm. Unlike the comparative study methods described in Chapter 7, these methods are specific to the RingBuilder algorithm. All tests are performed using the RingBuilder Interactive software described in Chapter 8. To validate the results from these tests, an independent test method is used to check the consistency of all design data and ensure that all design constraints are satisfied. For example, this program ensures that the flow on each ring does not exceed its capacity and is consistent with the flow implied by the route set. This program was instrumental in debugging all aspects of the design software. All tests are run on a PC equipped with 250 MBytes of RAM and an AMD Athalon processor running at 750 MHz.

The study is divided into three main subsections: (1) the unbalanced and balanced ring loading algorithms, (2) the demand packing heuristic and (3) the dithered sequencing heuristic. Note that the results for the unbalanced ring loading algorithm without demand packing or dithered sequencing are considered the *baseline* results throughout the remainder of this thesis. The specific test methods for each of these subsections is described in detail below.

9.2.1 Ring Loading Algorithms

The performance of the unbalanced and balanced ring loading algorithms are compared by generating network designs for all twelve combinations of test network and technology scenario described in Chapter 7. In the case of the balanced ring loading algorithm, the ADM discount factor and probabilistic selection exponent are set to 0.1 and 1.6, respectively. In addition to the total design cost and runtimes, a detailed list of design statistics is also presented for each network design.

We also characterize the effect of the ADM discount factor and the biased transport efficiency exponent on total design cost for the balanced ring loading algorithm. This is done by generating network designs for test cases 1 through 9 from Chapter 7 for a wide range of parameter values. In one set of tests, the ADM discount factor is swept over the range [0,1] in 0.1 increments, while holding the biased transport efficiency exponent constant at 1.6. Similarly, the biased transport efficiency exponent is tested for values ranging from 1.0 to 2.4 in 0.2 increments, while holding the ADM discount factor constant at 0.1. The results of these tests are presented in Section 9.3.1.

9.2.2 Demand Packing

To examine the effect of demand packing on total design cost, we generate network designs for all twelve test cases using both the unbalanced and balanced ring loading algorithms. In all cases, the demand packing ADM discount factor is set to 0.1 and demand packing is performed after each iteration. For the results using the balanced ring loading algorithm, the biased transport efficiency exponent is set to 1.6 and the (balanced ring loading) ADM discount factor is also set to 0.1.

We also examine the effect of the ADM discount factor (for demand packing) on the total design cost by generating a series of network designs for test cases 1 through 9. In these tests, the ADM discount factor is swept over the range [0,1] in 0.1 increments. The results of these tests are summarized in Section 9.3.2

9.2.3 Dithered Sequencing

Similar tests are also conducted for the Dithered Sequencing improvement heuristic. In all test cases, the distribution for the probabilistic selection method is limited to the top 10 ranked candidate rings. To obtain insights into the effect of the probabilistic selection method on total design cost, nine independent network designs are generated for test cases 1 through 9 using the probabilistic selection method and the balanced ring loading algorithm. In this suite of tests, the greedy selection threshold is set to 0.5 and the probabilistic selection exponent is set to 3.0. For the balanced ring loading algorithm, the ADM discount factor and the biased transport efficiency exponent parameters are set to 0.1 and 1.6, respectively.

Two sets of tests are also conducted using the unbalanced ring loading algorithm for a wide variety of parameters settings. The first set of tests considers the effect of the number of branching points, n, elite design sequences per branch, r, and branches per elite design sequence, k, on solution quality. For convenience, we denote these parameters by the 3-tuple (n,r,k). Parameters n and r are swept over the range [2,4] while holding the other parameters constant at $n,r,k = 3$. Similarly, parameter k is swept over the range [2,5] while holding the other two parameters $n,r = 3$. In these tests, the probabilistic selection exponent and greedy selection threshold were held constant at 3.0 and 0.9, respectively.

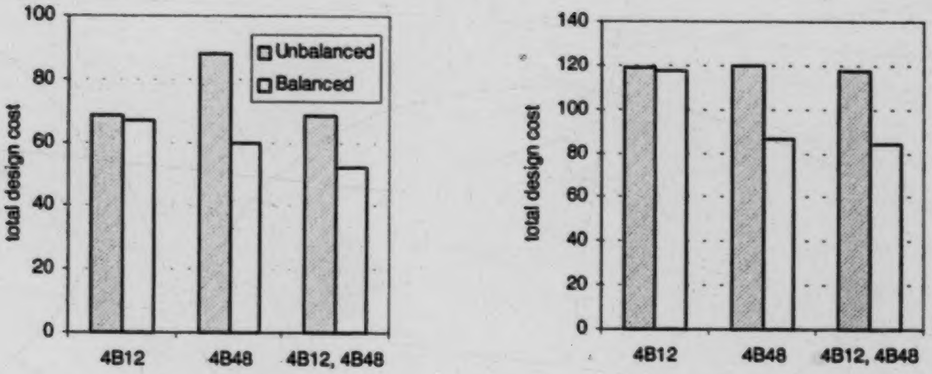
The second set of tests evaluates the effect of the probabilistic selection method parameters on the dithered sequencing solution quality for test case 5 (Net20, 4B192). First, the probabilistic selection exponent is held constant at 3.0 and the greedy selection threshold is swept over the range from 0.0 to 1.0, in 0.1 increments. Then the greedy selection threshold is set to 0.9 and the probabilistic selection exponent is tested for integer values ranging from 0 to 10. Both of these tests are conducted for two parameter settings: (3,3,3) and (1,9,1). Note that parameter setting (1,9,1) is equivalent to nine independent network designs because there is only one branching point. The results of all dithered sequencing tests are presented in Section 9.3.3.

9.3 Results

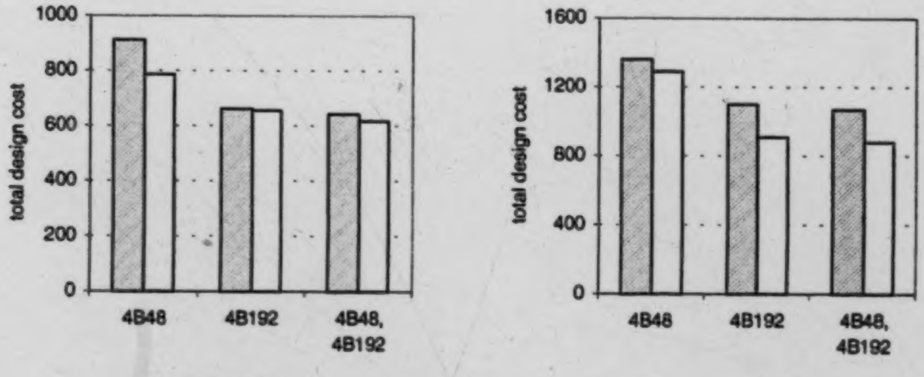
9.3.1 Ring Loading Algorithms

The results of the unbalanced and balanced ring loading algorithms are summarized in Fig. 9.1. Detailed statistics for the designs generated by these algorithms are also provided in Tables 9.1 and 9.2 at the end of this chapter. These results show that in all test cases the balanced ring loading algorithm generates the lowest cost design. The cost improvement ranges from less than 1% for test case 8 (Net32, 4B192) to up to 32% for test case 2 (Net15, 4B48). With the exception of Net32, the improvement due to the balanced ring loading algorithm actually increases with larger ring capacities. In general, this is because there is more slack available, on average, in rings with larger capacities. This improves the chances of successfully rerouting demands in the alternate direction around the ring, thereby reducing the number of rings and ADMs required in the design. In contrast, the unbalanced ring loading algorithm cannot take advantage of these opportunities. This is particularly noticeable in the metropolitan area networks (i.e., Net15 and Net20) where the total design costs for the 4B48 and the 4B48, 4B192 designs are either the same or higher than the 4B12 designs.

In the long-haul networks (i.e., Net32 and Net43), the total design costs for both ring loading algorithms decrease with larger ring capacities. The majority of this decrease is due to the reduction in the amount of fibre and regenerators required with larger ring capacities. For example, in Net32 the amount of fibre and regenerators drops by about 30-50% as the ring capacity quadruples from 4B48 to 4B192. Because the cost of the regenerators and fibre accounts for roughly 80% of the total, the decrease in total design cost is significant. Although the same reductions in the amount of fibre and regenerators occur in the metropolitan area networks, the effect is less significant because the distance dependent costs account for only 10% of the total.



(a) Net15 (b) Net20



(c) Net32 (d) Net43

Figure 9.1. Results for unbalanced and balanced ring loading algorithms.

The results in Tables 9.1 and 9.2 also show that runtimes increase dramatically with problem size (i.e., the number of candidates and demand pairs). For example, the runtimes for Net15 are in the range of 1/2 a minute to 2 minutes, whereas the runtimes for Net43 range from 1/2 to 2 hours. These results also show that the runtimes for the balanced ring loading algorithm are typically much longer than those for the unbalanced ring loading algorithm. For example, the runtimes for the balanced ring loading algorithm are up to 4 times longer than the unbalanced ring loading algorithm for Net43. The main reason for this difference is that the balanced ring loading algorithm handles many more routes in loading each candidate ring.

Examples of the network designs generated for test case 2 (i.e., Net15, 4B48) using the unbalanced and balanced ring loading algorithms are shown in Fig. 9.2 and Fig. 9.3, respectively.

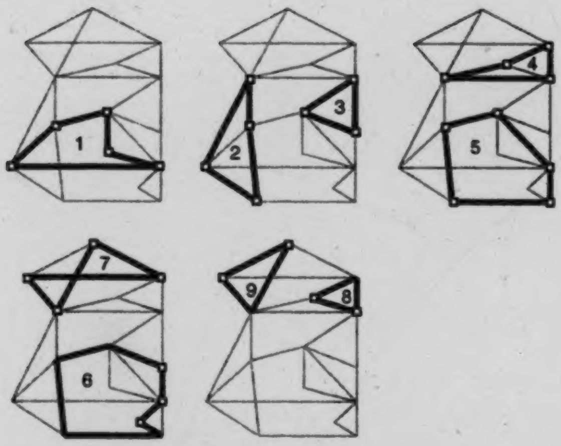


Figure 9.2. Network design for test case 2 (Net15, 4B48) using the unbalanced ring loading algorithm.

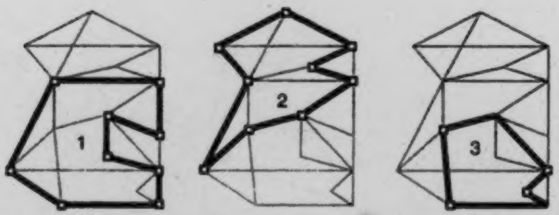


Figure 9.3. Network design for test case 2 (Net15, 4B48) using the balanced ring loading algorithm.

These examples illustrate typical differences in the network designs generated using the two ring loading algorithms. The first and most obvious difference is the number of rings in each design. The designs generated using the balanced ring loading algorithm generally require about half the number of rings as those generated using the unbalanced ring loading algorithm. This is due primarily to the ability of the balanced ring loading algorithm to alter the routing of demands during the loading process.

Although the re-routing of demands tends to increase the average route length and the average working load per span, the total network capacity in the balanced ring loading designs is actually lower in most cases. The results in Tables 9.1 and 9.2 also show that the balanced ring loading designs require half as many inter-ring transitions, on average, as the unbalanced ring loading designs. This is due in part to the dynamic routing capability of the balanced ring loading algorithm, which forces the route to be carried along on each ring between the farthest entry and exit nodes. In contrast, the unbalanced ring loading algorithm only carries the route along the spans that intersect the ring. As a result, the a single route can enter and exit the same ring more than once. Furthermore, the order in which route segments are loaded by the balanced ring loading algorithm depends on the number of inter-ring transitions. That is, all other things being equal, the route segments with fewer inter-ring transitions are loaded first. The unbalanced ring loading algorithm does not account for this during the loading process. Another factor that influences the number of inter-ring transitions is the length of the rings in the design. The average number of nodes in the balanced ring loading designs is generally much higher than in the unbalanced ring loading designs. In test case 2, for example, the average number of nodes per ring for the unbalanced ring loading design is only 3.17, whereas the average number of nodes per ring for the balanced ring loading design is 8. Clearly, larger rings (i.e., with more nodes) have a greater potential to carry demands farther along their route from origin to destination, thereby minimizing the number of inter-ring transitions.

Another difference in the designs generated with the two loading algorithms is the number spans covered by the ring set. In general, the designs generated with the balanced ring loading cover fewer spans in the network topology. In test case 2, for example, the balanced ring loading design covers only 21 of 28 spans. The unbalanced ring loading design, on the other hand, covers all 28 spans because the initial working load on these spans is greater than zero. Thus, the dynamic routing capabilities of the balanced ring loading algorithm overcomes an inherent disadvantage of the basic span coverage approach by allowing spontaneous *span eliminations* to occur as a result of the ring loading process. It is well known that span eliminations can significantly reduce total network design costs [Fla90], [Lee00]. All of the methods proposed to date for span elimination, however, rely on topology reduction techniques. Therefore, the balanced ring loading algorithm represents an alternative to these more direct approaches.

For the balanced ring loading algorithm, the effect of the ADM discount factor on the total design cost for test cases 5 (Net20, 4B48) and 6 (Net20, 4B48 & 4B192) is shown in Fig. 9.4. In general, this figure shows that the best designs are obtained with an ADM discount factor in the range from 0.4 to 0.7 and at 0.1. These results are representative of the other test cases, except test cases 8 (Net32, 4B192) and 9 (Net32, 4B48 & 4B192) where the total design cost is insensitive to changes in the ADM discount factor. This is due to the hubbed demand pattern in Net32. With a hubbed demand pattern, every route segment that may be loaded onto a candidate ring after the first iteration requires exactly one ADM to be placed. Therefore, the ADM discount factor has no effect on the order in which route segments are loaded.

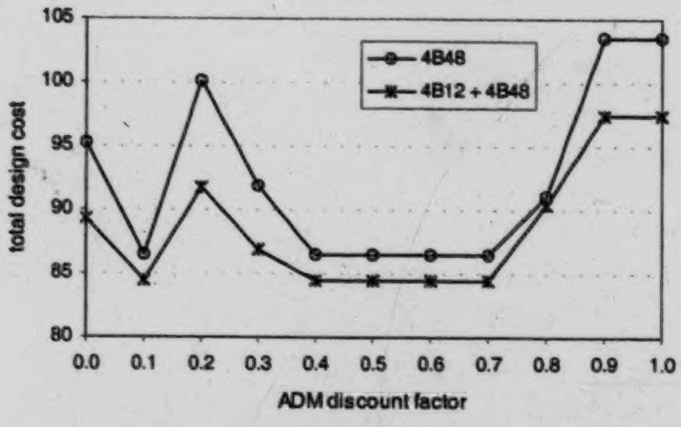


Figure 9.4. Effect of the ADM discount factor on total design cost for test case 5 (Net20, 4B48) and test case 6 (Net20, 4B48 & 4B192) using the balanced ring loading algorithm.

The effect of the biased transport efficiency exponent on total design cost for test cases 5 (Net20, 4B48) and 6 (Net20, 4B12 & 4B48) is shown in Fig. 9.5. Note that an exponent value $a = 1.0$ is equivalent to the baseline case in which rings are selected purely on the transport efficiency metric. For the multi-technology design (test case 6), this figure shows that the optimum exponent value lies within the range $a = [1.6, 1.8]$. Within this range, the total design cost is roughly 14% lower than the baseline case. The main reason for this cost improvement is a marked reduction in the number of 4B12 rings. In the baseline design (generated with an exponent value equal to 1.0), there are nine 4B12 rings and two 4B48 rings. In contrast, the design generated with an exponent value of 1.6 has only one 4B12 ring and three 4B48 rings. Similar results are obtained for the other multi-technology test cases. This demonstrates the advantage of biasing the selection metric to favour rings with larger flow-distance products in multi-technology designs.

The best designs for the single technology design (test case 5) are also obtained with an exponent value within the range $a = [1.6, 1.8]$, however, the results are only marginally better than the baseline. These results can be explained by the fact that there are no economy-of-scale effects to leverage in single technology designs. Therefore, the only advantage that biasing the selection metric offers is the possibility of placing fewer rings in total. For example, in the baseline design for test case 5 there are five 4B48 rings, whereas in the design generated with an exponent value of 1.6 there are only four 4B48 rings. The results for the other single technology test cases are similar.

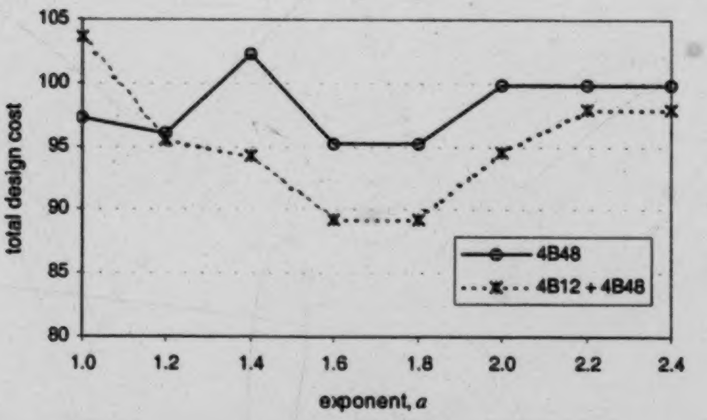
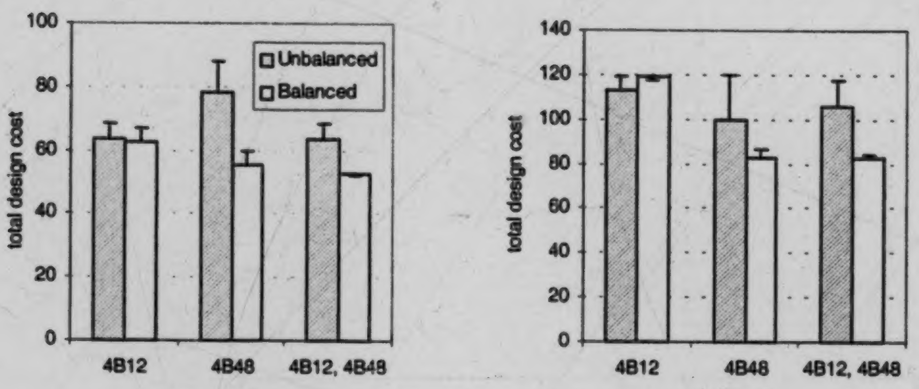


Figure 9.5. Effect of the biased transport efficiency exponent on total design cost for test case 5 (Net20, 4B48) and test case 6 (Net20, 4B12 & 4B48) for Net20 using the balanced ring loading algorithm.

9.3.2 Demand Packing

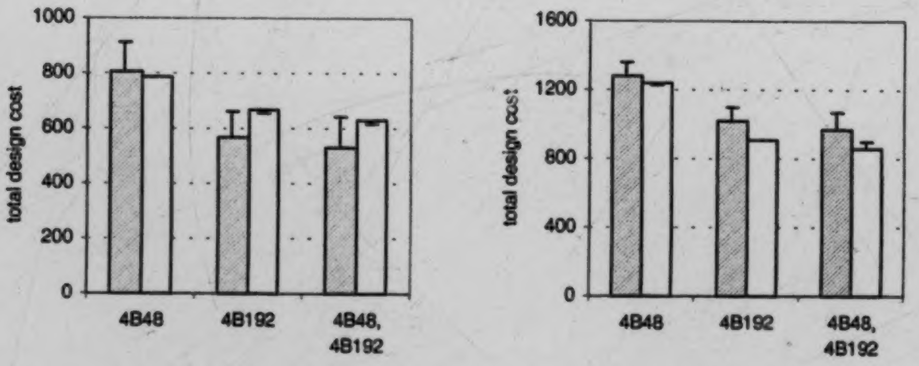
Results for unbalanced and balanced ring loading algorithms with demand packing are summarized in Fig. 9.6. The total design costs without demand packing are also indicated in Fig. 9.6 by the vertical error bars. Detailed statistics for the designs generated by these algorithms are also provided in Tables 9.3 and 9.4 at the end of this chapter.

The results show that demand packing provides a significant improvement in design cost for the unbalanced ring loading algorithm. The improvement in design costs is from 5% for test case 4 (Net20, 4B12) to up to 18% for test case 9 (Net32, 4B48 & 4B192). On average, the demand packing algorithm provides a 10% reduction in total design cost for the unbalanced ring loading algorithm. In three of the test cases, the cost of the unbalanced ring loading algorithm solutions (with demand packing) are actually lower than those obtained with the balanced ring loading algorithm.



(a) Net15

(b) Net20



(c) Net32

(d) Net43

Figure 9.6. Results for unbalanced and balanced ring loading algorithms with demand packing.

The demand packing results for the balanced ring loading algorithm are somewhat mixed. In more than half of the test cases, the total design cost actually increases (by up to 1.8%) as a result of demand packing. In the remaining test cases, the improvement ranges from a couple percent up to 7%. One explanation for these results is that the average ring fill in the balanced ring loading solutions is generally much higher in the first place, so there is less slack in which to pack unserved demands. In addition, in those cases where the cost increases the designs are generally quite similar except the designs generated with demand packing contain slightly more ADMs. This suggests that better results may be obtained with a higher (demand packing) ADM discount factor.

In almost all test cases, the total runtime is also reduced by demand packing. This is because fewer rings are generally required with demand packing and the time required for demand packing is small in comparison with the time required to evaluate and select additional candidate rings.

Figure 9.7 shows the effect of the (demand packing) ADM discount factor on total design cost for test cases 5 and 6 (Net20). These results are representative of the other test cases. In almost all test cases, the total design cost decreases with increasing values of ADM discount factor up to a value of 0.3. Beyond this point, the total design cost remains unchanged in all test cases. In two test cases for Net32, the total design cost is completely insensitive to changes in the ADM discount factor. A closer inspection of the results showed that none of the demands that were packed required an ADM to be placed along the route. In this situation, the ADM discount factor does not have an effect on the outcome.

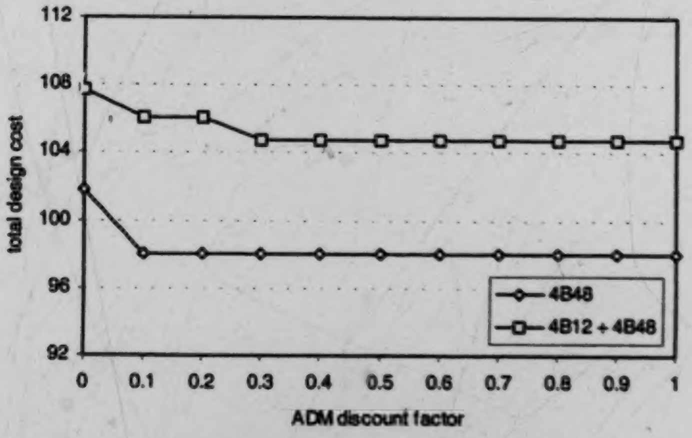


Figure 9.7. Total design cost versus demand packing ADM discount factor for test case 5 (Net20, 4B48) and 6 (Net20, 4B12 & 4B48).

Overall, the results show that the best designs are obtained when there is a large cost penalty (i.e., ADM discount factor) associated with placing an ADM during demand packing. This suggests that relatively few other demands actually benefit from the placement of an ADM during demand packing. Therefore, the fixed cost of placing an ADM is incurred primarily by the current demand being packed. In this situation, it becomes more attractive (economically) to route demands over longer routes that do not require additional ADMs. While this increases the average number of hops per demand, the results show that the total design cost is marginally lower.

An alternative to using an ADM discount factor would be to establish a flow threshold for placing an ADM. That is, if the maximum flow on a route exceeds a given threshold, the cost of placing an ADM is set to zero, otherwise the full cost of the ADM is used in finding the least-cost path. This alternative is left as an item for future research.

9.3.3 *Dithered Sequencing*

The effect of the greedy selection threshold on total design cost for test case 5 (Net20, 4B48) is summarized in Fig. 9.8. For each value of the greedy selection threshold, nine designs are generated using the probabilistic selection method with an exponent value $x = 3.0$. The minimum, maximum and mean costs for these designs are plotted in Fig. 9.8.

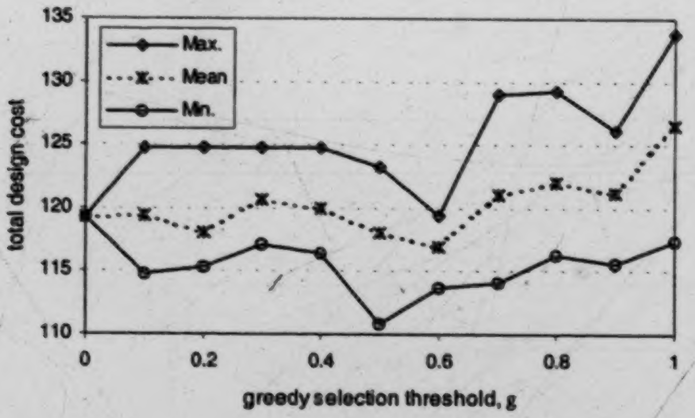


Figure 9.8. Total design cost versus greedy selection threshold for test case 5.

Note that a greedy selection threshold $g = 0$ corresponds to the basic greedy selection method and, hence, all nine designs are identical. At the other extreme, a value $g = 1$ represents the case where all ring choices are made using the probabilistic selection method. These results show that the best solution is obtained with a greedy selection threshold $g = 0.5$. At progressively higher values, the total design costs tend to rise. This occurs because comparatively poor ring choices are made at the end of the design, despite the adaptive behaviour of the selection method.

Figure 9.9 shows the total design cost as a function of the probability selection exponent for test case 5 with the greedy selection threshold held constant at $g = 0.5$. This figure shows that the minimum design cost is achieved with an exponent value $x = 3$. Note that an exponent value of $x = 0$ corresponds to a uniform probability distribution, i.e. there is an equal probability of selecting any one of the top ten candidate rings. As the exponent value increases, it becomes increasingly likely that the top ranked candidate will be selected. The results indicate, however, that there is still a large degree of variability in the total design costs even at higher exponent values.

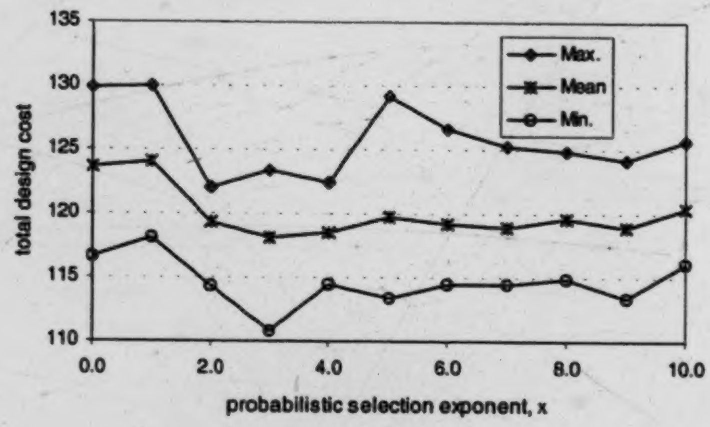
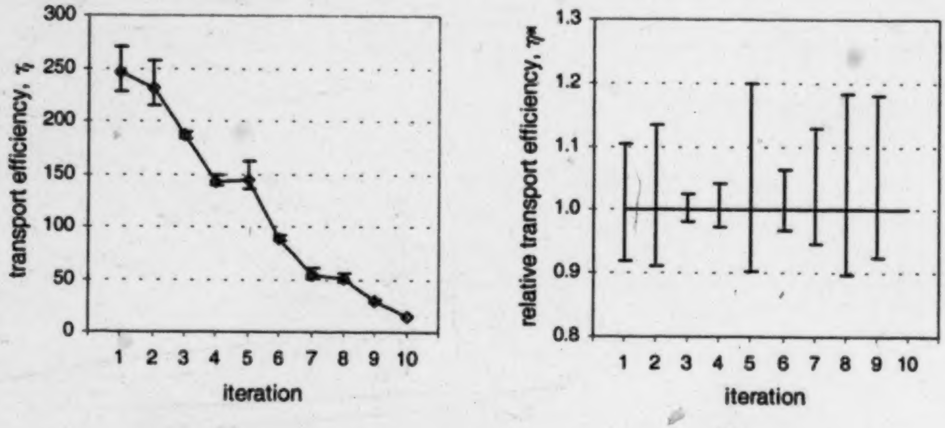


Figure 9.9. Design cost versus probabilistic selection exponent for test case 5 with a greedy selection threshold $g = 0.5$.

For further insight into this outcome, we plot the transport efficiencies for the top ten candidate rings for each iteration is the design process in Fig. 9.10. Figure 9.10(a) shows the minimum, maximum and mean transport efficiencies for the elite candidate rings at each iteration. Figure 9.10(b) shows the range in transport efficiencies relative to the mean. These plots show that while there is an order of magnitude difference in the mean transport efficiency (of the elite candidate rings) between the first and final iterations, there is only a small difference in the range between the minimum and maximum transport efficiencies relative to the mean.

As a result, the probability distribution for the elite candidate rings is not significantly different between the first and final iterations, as shown in Fig. 9.11 and Fig. 9.12. These figures show the probability of selecting the k^{th} ranked candidate ring for several exponent values. Even with higher exponent values, there is still a high probability of selecting a lower ranked candidate ring. For example, in the final iteration there is a 63% chance that the highest ranked ring will not be selected with an exponent value $x = 10$.

The relatively small difference in the transport efficiencies amongst the top-ranked candidate rings also helps to explain the results of the previous tests on the greedy selection threshold (i.e., the benefit of reverting to the greedy selection method after a certain percentage of the demand is served). An alternative to this would be to introduce a simulated annealing (SA)-like "cooling schedule" into the probabilistic selection method that gradually increases the greediness of the selection process. This is a possible item for future research.



(a) (b)

Figure 9.10. Plot of the (a) absolute and (b) relative minimum, maximum and mean transport efficiencies of the elite candidate rings as a function of design iteration.

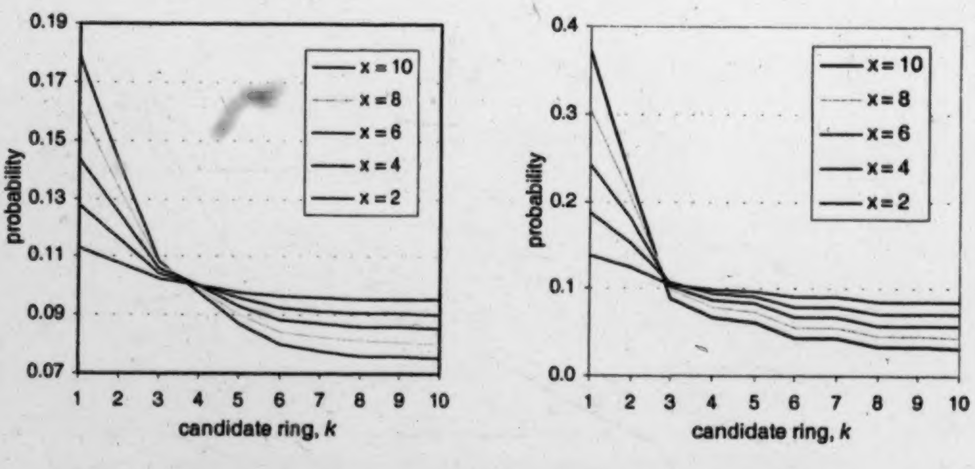


Figure 9.11. Probability distribution of the top ten candidate rings as a function of the probabilistic selection exponent *x* for the (a) first and (b) final iterations.

The results for all designs generated with the probabilistic selection method and balanced ring loading are summarized by the histogram in Fig. 9.12. This histogram shows the relative frequency distribution of the total design cost (relative to the greedy solution) for all 81 designs generated in this suite of tests. These results show that while the majority of designs generated with the probabilistic selection method have a total design cost greater than the baseline solution, improvements of up to 10% are attainable by simply generating multiple designs for the same planning problem. Indirectly, these results also suggest that the greedy selection method produces solutions that are generally within about 10% to 12% of the best solutions that can be obtained using the balanced ring loading method.

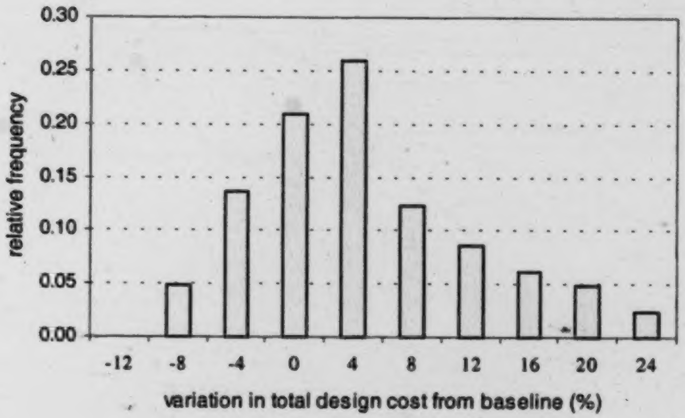


Figure 9.12. Relative frequency histogram of the total design cost (relative to the baseline greedy solution) for test cases 1-9 using the probabilistic selection method and the balanced ring loading algorithm.

Up to this point, the results have focused on the efficacy of the probabilistic selection method on independent designs. We now present the results for the dithered sequencing meta-heuristic. A plot of the progress of a typical dithered sequencing design is shown in Fig. 9.13. Note that at each branching point, three new design sequences are spawned from each of three of the design sequences. This figure also shows that the percentage of demand served by each design sequence varies substantially at each branching point.

The effect of different dithered sequencing parameter settings on total design cost are shown in Fig. 9.14 and 9.15. These figures show the range in the solution values for all design generated with each parameter setting, not just the minimum values. Figures 9.14(a) and 9.14(b) show that the minimum, mean and maximum solution values all decrease when either the number of elite sequences per branching point, *r*, or the number of branches per elite sequence, *k*, increases. This can be explained by the fact that more design sequences are generated in total as value of these parameters increase. Because there are more design sequences to choose from at each branching point, the subset of elite design sequences are likely to have lower solution values in general.

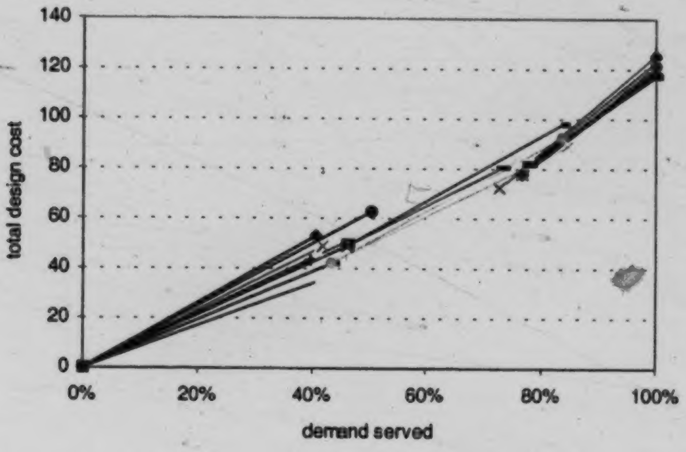
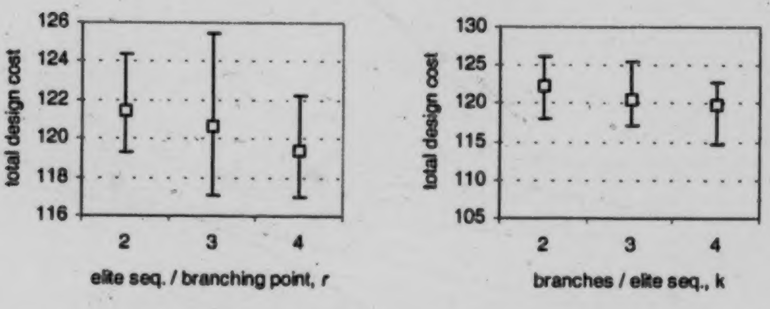


Figure 9.13.Design sequences for test case 5 using parameter settings (3,3,3).



(a)

(b)

Figure 9.14. Effect of different Dithered sequencing parameter settings on total design cost (a) no. of branching points and (b) no. of elite sequences per branching point.

In contrast, the minimum, maximum and mean solution values do not decrease monotonically with increases in the number of branching points, as shown in Fig. 9.15. The range between the minimum and maximum values decreases significantly, however, when the number of branching points is set to five. This occurs because there may be only one or two rings added between branching points and, therefore, there is less opportunity for the design sequences to deviate from the best solutions at each branching point.

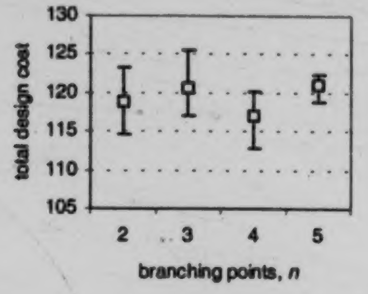


Figure 9.15. Effect of no. of branches per elite sequence on total design cost.

Because multiple design sequences are evaluated at each stage of the design process, the total runtime for each dithered sequencing run is roughly proportional to the number of design sequences times the runtime of the a single greedy solution. For example, the runtime for the default parameter setting (3,3,3) is about nine times longer than the runtime for the basic greedy solution.

A surprising outcome of these tests is that the best (i.e., lowest cost) design is actually obtained when each design sequence is completely independent. This outcome can be explained with the aid of Fig. 9.8 and Fig. 9.16, which shows the total design cost as a function of the greedy selection threshold for test case 5 for parameter settings (3,3,3).

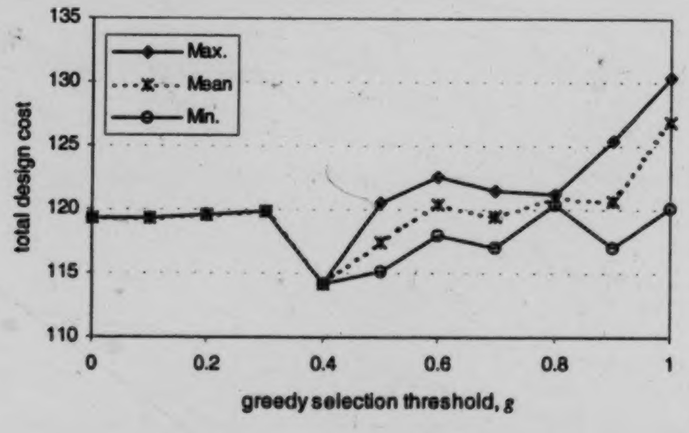


Figure 9.16. Total design cost versus greedy selection threshold for test case 5 with parameter setting (3,3,3).

Figure 9.16 shows that for the case where Dithered Sequencing is used, the solution costs are tightly clustered about the mean over the entire range in the greedy selection threshold. In contrast, Fig. 9.8 shows that the deviation in total design cost for the independent solutions is much greater both above and below the mean. This observation can be explained by the fact that the pruning procedure limits the search to a subset of elite design sequences at each branching point. Because the mean values are roughly the same over the entire range, however, it is more likely that a lower cost solution will be found by generating several independent designs using the probabilistic selection method. These results indicate that the measure used to prune design sequences at each branching point is not always a good predictor of the actual merit of the design sequence. While it clearly excludes consideration of inferior design sequences (as evidenced by the higher maximum values in the independent results), it also abandons the search of design sequences that eventually prove worthwhile.

9.4 Summary

Overall the experimental results in this chapter show that significant savings in total design cost can be achieved relative to the basic greedy solution using several of the improvement heuristics developed in this chapter. Of these heuristics, both the balanced ring loading and demand packing algorithms provide the greatest improvement with savings of up to 32% relative to the benchmark solutions. The results from the dithered sequencing heuristic were less encouraging. Nonetheless, the probabilistic selection method provides a useful means of generating several alternative solutions for the same design problem and may be used with any of the improvement heuristics proposed here.

In the next chapter, we consider Mathematical Programming formulations for the same problems that offer further evidence that the solutions generated by these heuristics are very good indeed.

Table 9.1: Design Statistics for Unbalanced Ring Loading Algorithm

Test Case	Net-work	Number of Rings			Number of ADMs			Regen.	Fibre (km)	Avg. Fill (%)	Inter-ring Transit.	Avg. Hops/ Route	Spans Elim.	Total Working (DS3-hops)	Total Capacity (DS3-hops)	Total Cost	Runtime (sec.)
		4B12	4B48	4B192	4B12	4B48	4B192										
1	Net15	12	-	-	47	-	-	11	902.0	53.7	47	1.82	0	374	696	68.71	45
2	"	-	9	-	-	32	-	6	591.8	20.5	94	1.82	0	374	1,824	87.86	26
3	"	12	0	-	47	0	-	11	902.0	53.7	47	1.82	0	374	696	68.71	78
4	Net20	19	-	-	71	-	-	22	2,333	62.0	143	1.99	0	692	1,116	118.77	69
5	"	-	10	-	-	38	-	9	1,182	30.7	182	1.99	0	692	2,256	119.31	37
6	"	15	3	-	46	12	-	20	1,928	50.6	167	1.99	0	692	1,368	117.74	105
7	Net32	-	13	-	-	65	-	377	126,444	43.1	548	5.32	1	1,882	4368	910.12	23
8	"	-	-	10	-	-	47	199	70,616	20.0	621	5.32	2	1,882	9,408	660.68	15
9	"	-	7	4	-	25	24	215	75,316	30.4	621	5.32	2	1,882	6,192	645.38	26
10	Net43	-	47	-	-	223	-	214	94,764	65.7	2,692	3.52	4	8,890	13,536	1,358.1	1,610
11	"	-	-	26	-	-	110	89	42,160	35.6	3,025	3.52	5	8,890	24,960	1,097.4	758
12	"	-	22	8	-	83	56	96	49,176	56.6	2,927	3.52	5	8,890	15,696	1,074.0	1,607

Table 9.2: Design Statistics for Balanced Ring Loading Algorithm

Test Case	Net-work	Number of Rings			Number of ADMs			Regen.	Fibre (km)	Avg. Fill (%)	Inter-ring Transit.	Avg. Hops/ Route	Spans Elim.	Total Working (DS3-hops)	Total Capacity (DS3-hops)	Total Cost	Runtime (sec.)
		4B12	4B48	4B192	4B12	4B48	4B192										
1	Net15	6	-	-	46	-	-	8	895.6	69.9	42	2.20	0	453	648	66.58	106
2	"	-	3	-	-	22	-	2	373.4	44.3	33	2.48	7	510	1,152	59.87	50
3	"	3	1	-	18	9	-	3	449.7	69.9	34	2.32	4	478	684	52.26	104
4	Net20	10		-	77	-	-	32	2,819	73.5	23	2.76	1	961	1,308	117.20	130
5	"		4	-	-	29	-	11	1,066	59.7	35	3.30	3	1,147	1,920	86.43	51
6	"	1	3	-	2	27	-	11	1,066	64.6	35	3.30	3	1,147	1,776	84.43	99
7	Net32	-	8	-	-	53	-	345	115,396	55.8	172	5.91	2	2,091	3,744	786.88	24
8	"	-	-	6	-	-	45	226	77,596	18.2	250	5.63	3	1,993	10,944	655.88	19
9	"	-	6	1	-	31	16	234	80,480	40.3	250	6.23	3	2,207	5,472	617.90	31
10	Net43	-	32	-	-	220	-	230	99,732	72.6	2,127	3.71	1	9,374	12,912	1,288.7	6,552
11	"	-	-	13	-	-	92	70	35,920	53.7	2,193	4.04	12	10,200	19,008	907.15	2,932
12	"	-	5	8	-	21	71	77	37,160	64.2	2,237	3.98	13	10,042	15,648	877.15	5,484

Table 9.3: Design Statistics for Unbalanced Ring Loading Algorithm with Demand Packing

Test Case	Net-work	Number of Rings			Number of ADMs			Regen.	Fibre (km)	Avg. Fill (%)	Inter-ring Transit.	Avg. Hops/ Route	Spans Elim.	Total Working (DS3-hops)	Total Capacity (DS3-hops)	Total Cost	Runtime (sec.)
		4B12	4B48	4B192	4B12	4B48	4B192										
1	Net15	10	-	-	41	-	-	8	770.6	71.4	67	2.04	2	420	588	63.45	40
2	"	-	7	-	-	27	-	2	445.3	32.2	113	2.17	3	448	1,392	78.23	26
3	"	10	0	-	41	0	-	8	770.6	71.4	67	2.04	2	420	588	63.45	72
4	Net20	17	-	-	66	-	-	15	1,960	79.0	168	2.21	1	768	972	113.00	63
5	"	-	6	-	-	29	-	5	827.2	45.6	193	2.14	4	745	1,632	98.04	28
6	"	10	3	-	39	12	-	8	1,341	67.4	174	2.21	2	768	1,140	106.11	78
7	Net32	-	10	-	-	58	-	325	110,224	54.0	537	5.57	1	1,971	3,648	803.52	26
8	"	-	-	9	-	-	42	161	57,604	23.8	628	5.67	5	2,008	8,448	568.72	25
9	"	-	5	4	-	18	24	161	57,604	35.4	628	5.71	5	2,020	5,712	532.72	31
10	Net43	-	39	-	-	209	-	165	81,364	80.9	3,007	3.69	5	9,321	11,520	1,283.6	1,655
11	"	-	-	21	-	-	94	69	34,028	50.7	3,331	4.13	16	10,425	20,544	1,019.1	659
12	"	-	15	8	-	58	56	74	39,228	69.5	2,965	3.90	10	9,839	14,160	985.97	1,408

Table 9.4: Design Statistics for Balanced Ring Loading Algorithm with Demand Packing

Test Case	Net-work	Number of Rings			Number of ADMs			Regen.	Fibre (km)	Avg. Fill (%)	Inter-ring Transit.	Avg. Hops/ Route	Spans Elim.	Total Working (DS3-hops)	Total Capacity (DS3-hops)	Total Cost	Runtime (sec.)
		4B12	4B48	4B192	4B12	4B48	4B192										
1	Net15	6	-	-	44	-	-	5	841.9	71.3	49	2.03	0	419	588	62.41	87
2	"	-	3	-	-	20	-	1	308.2	50.9	34	2.49	9	513	1,008	55.44	46
3	"	3	1	-	18	9	-	3	449.7	70.3	36	2.33	4	481	684	52.75	100
4	Net20	10		-	78	-	-	26	2,561	79.8	55	2.86	1	996	1,248	118.90	124
5	"		3	-	-	27	-	9	1,022	69	41	3.43	5	1,192	1,728	82.41	49
6	"		3	-	0	27	-	9	1,022	69	41	3.43	5	1,192	1,728	82.41	90
7	Net32	-	9	-	-	53	-	346	115,464	55.2	171	5.92	2	2,094	3,792	787.32	27
8	"	-	-	7	-	-	45	234	79,572	17.6	251	5.62	3	1,989	11,328	667.46	18
9	"	-	7	1	-	31	16	242	82,456	39.5	251	6.22	3	2,202	5,568	629.48	32
10	Net43	-	27	-	-	210	-	188	88,684	83.5	2,171	3.81	3	9,619	11,520	1,244.37	6,520
11	"	-	-	12	-	-	88	78	36,416	56.0	2,334	4.17	14	10,540	18,816	909.33	2,980
12	"	-	7	7	-	24	65	76	35,644	73.7	2,318	4.25	14	10,721	14,544	859.47	5,478

Table 9.5: Design Statistics for Balanced Ring Loading Algorithm with Dithered Sequencing

Test Case	Net-work	Number of Rings			Number of ADMs			Regen.	Fibre (km)	Avg. Fill (%)	Inter-ring Transit.	Avg. Hops/ Route	Spans Elim.	Total Working (DS3-hops)	Total Capacity (DS3-hops)	Total Cost	Runtime (sec.)
		4B12	4B48	4B192	4B12	4B48	4B192										
1	Net15	6	-	-	42	-	-	7	816.0	70.9	37	2.02	1	417	588	61.48	954
2	"	-	3	-	-	22	-	1	322.8	50.5	23	2.71	9	558	1,104	58.41	450
3	"	2	1	-	14	10	-	4	382.1	72.8	31	2.46	6	507	696	50.11	936
4	Net20	12	-	-	74	-	-	28	2,562	74.0	41	2.60	0	906	1,224	113.91	1,170
5	"	-	4	-	-	30	-	11	1,091	60.0	19	3.39	4	1,180	1,968	86.96	459
6	"	3	2	-	10	22	-	19	1,315	69.4	19	3.16	1	1,099	1,584	83.68	891
7	Net32	-	9	-	-	54	-	293	99,788	62.3	264	6.08	2	2,152	3,456	709.64	216
8	"	-	-	5	-	-	37	219	73,792	22.5	171	6.47	4	2,290	10,176	595.56	171
9	"	-	6	1	-	22	16	238	78,844	39.1	159	6.57	5	2,326	5,952	583.42	279
10	Net43	-	33	-	-	220	-	241	101,384	71.0	2,192	3.74	1	9,436	13,296	1,340.6	58,968
11	"	-	-	12	-	-	89	74	35,924	52.7	2,148	3.89	11	9,824	18,624	891.47	26,388
12	"	-	7	8	-	25	74	89	41,024	61.1	2,253	4.12	12	10,406	17,040	920.47	49,356

10 Research on Mathematical Programming applied to Multi-Ring Network Design

10.1 Introduction

The heuristic algorithms developed in the previous chapter provide approximate and apparently quite efficient (i.e., near-optimal) solutions for a relatively accurate model of the multi-ring network design problem that captures many of the finer details in the real-world planning problem. In this chapter, we present three mathematical programming formulations that offer optimal solutions (within computational constraints) but for more approximate models of the problem. That is, the formulations presented here adopt several simplifying assumptions that make the problem more tractable for optimal solution methods. Each of these formulations represents a different trade-off between model detail and tractability. Each, however, provides certain utility and insight into the overall problem in its full detail.

All three formulations developed here are expressed as integer (linear) programs (IPs). These formulations differ from the work in [KNR97], [BTR96] and [CQT96] (previously surveyed in Chapter 6) by modelling aspects such as modularity, inter-ring routing costs and, in some cases, glass-through locations. While the proposed formulations have certain claims to optimality within the logical problem model, computational constraints may prohibit finding strictly optimal solutions for large problems. Nonetheless, even when solved with relaxed tolerances on optimality, they can provide very good, feasible, designs of significant real-world size that also serve as benchmarks for heuristic solutions.

The remainder of this chapter is organized as follows. We begin by introducing the mathematical notation used to express the IP formulations. The formulations are then developed in Sections 10.3 through 10.5. Section 10.6 describes the study method used to evaluate the performance of these formulations and the results are given in Section 10.7. This is followed by a summary and interpretation of the findings in Section 10.8.

10.2 Notation

The notation used in subsequent sections is defined below.

Sets:

— J: set of candidate rings.
— I: set of routes.
— K: set of demands.
— $I(k)$: subset of routes available for demand k.
— $I(s)$: subset of routes that intersect span s.

— $J(i)$: subset of ring candidates that intersect route i.

— $J(s)$: subset of ring candidates that cover span s.

— $S(i)$: subset of spans in route i.

— $S(j)$: subset of spans in candidate ring j.

Design Parameters:

— α_{jk}: equals one if demand k is carried (in part or whole) by ring j and zero otherwise (used in section 10.4 only).

— b_{ij}: unit ADM and cross-connect add-drop interface costs associated with carrying route i on ring j.

— c_j: fixed cost of ring j.

— d_k: size (quantity) of demand bundle k.

— m_j: capacity of ring candidate j.

— w_s: working load on span s.

Decision Variables:

— X_j: integer number of ring candidate j to instantiate.

— G_{ij}: quantity of demand on route i carried by ring candidate j.

— F_i: quantity of demand carried on route i.

10.3 Multi-Modular Pure Span Coverage (SCIP)

In the first formulation, we assume that the demands have already been routed over the network topology and the working load on each span w_s is given along with a set of candidate BLSR rings J. Associated with each candidate $j \in J$ is a capacity m_j and a list of spans $S(j)$ that it traverses. It is also assumed that all nodes in each candidate ring are equipped with an ADM and there are no constraints on where inter-ring transitions occur. Based on these simplifying assumptions, the objective of the problem is to find a min-cost set of rings whose aggregate capacity covers the working load assigned on all spans. This is a special case of the *Span Coverage* problem described in Chapter 6. The problem can be stated algebraically as:

SCIP

Minimize:

$$\sum_{j \in J} c_j \cdot X_j \qquad (10.1)$$

Subject to:

$$\sum_{j \in J(s)} m_j \cdot X_j \geq w_s \qquad \forall s \in S \qquad (10.2)$$

$$X_j \geq 0 \text{, integer,} \qquad \forall j \in J \qquad (10.3)$$

In the above formulation, the cost c_j of candidate ring j includes all fixed costs involved in establishing the ring (e.g., fibre and ADM common equipment) but excludes variable costs such as add-drop interfaces and inter-ring transitions. The set of ring candidates, J, is generated by enumerating all cycles in the network graph and instantiating a candidate ring for each combination of cycle and ring capacity under consideration. In practice, the number of candidate rings can be restricted by imposing a limit on the number of nodes in the ring or on its physical distance. The objective function (10.1) is to minimize the total facility cost over all rings. Constraint set (10.2) ensures that the aggregate span capacity of all rings that intersect span s is greater than or equal to the working load assigned to that span from the initial demand routing. The total number of variables in this formulation is equal to the number of ring candidates and the total number of constraints is equal to the number of spans.

Note that this formulation is suitable for BLSR rings only because the capacity of an UPSR ring is shared amongst all its spans and, therefore, cannot be modelled on a per-span basis, as in constraint set (10.2). In addition, because each node in a ring is assumed to be equipped with an ADM, any real-world constraints on the number of ADMs in a ring may be violated by the solution. Another property of the span coverage approach is that the resultant design cost may depend heavily on the pre-determined demand routing pattern, as discussed in Chapter 6. Nonetheless, the SCIP formulation is a highly useful formulation for several purposes. In particular, it has been used extensively by Lee et al. [LGM99], [Lee00] for characterizing the comparative effects of various topology changes. This work shows that design costs can vary by up to 28% in this class of design by eliminating some spans from the network topology and making the necessary alterations to the routing pattern. The results of that work also show a very strong correlation (0.93) between the SCIP solutions and those obtained by the RingBuilder algorithm described in [Sle99]. High correlation with RingBuilder (though not the same high absolute accuracy in predicting design cost) means that SCIP can be conveniently used as a surrogate for more detailed design solutions in comparative studies.

10.4 Fixed Charge and Routing (FCRIP)

The next formulation aims to simultaneously determine both the ring set and the demand routing pattern and include the costs of ring-to-ring transitions. This is intended to address the limitations associated with fixed demand routing, as noted in the previous section. Like the previous formulation, we assume that a set of candidate BLSR rings J is given. Associated with each candidate $j \in J$ is a capacity m_j and a list of spans $S(j)$ that it traverses. It is also assumed that all nodes in each candidate ring are equipped with an ADM and there are also no constraints on where inter-ring transitions occur. A demand matrix is given along with a set of routes for each demand bundle. Based on these assumptions, the problem is to find a set of rings, the flow along each route and the assignment of flows to rings that minimizes the total cost of the design. We refer to this as the *Fixed Charge and Routing* problem, which can be stated algebraically as:

FCRIP

Minimize:

$$\sum_{j \in J} c_j \cdot X_j + \sum_{i \in I} \sum_{j \in J(i)} b_{ij} \cdot G_{ij} \tag{10.4}$$

Subject to:

$$\sum_{i \in I(k)} F_i = d_k, \qquad \forall k \in K \tag{10.5}$$

$$\sum_{j \in J(s)} G_{ij} \geq F_i, \qquad \forall i \in I, \forall s \in S(i) \tag{10.6}$$

$$\sum_{i \in I(s)} G_{ij} \leq m_j \cdot X_j, \qquad \forall j \in J, \forall s \in S(j) \tag{10.7}$$

$$X_j \geq 0 \text{, integer,} \qquad \forall j \in J \tag{10.8}$$

$$F_i \geq 0 \text{, integer,} \qquad \forall i \in I \tag{10.9}$$

$$G_{ij} \geq 0 \text{, integer,} \qquad \forall i \in I, \forall j \in J(i) \tag{10.10}$$

In the above formulation, the total cost is modelled by both a fixed ring establishment cost (e.g., ADM chassis) and variable routing cost, such as add-drop interfaces for inter-ring transitions. As in the previous formulation, the set of candidate rings J is generated by finding all cycles in the network graph and instantiating a candidate ring for all combinations of cycle and ring capacity under consideration. The routes $I(k)$ for each demand k are also generated by finding a subset of routes between the origin and destination nodes of the demand. In practice, it is usually necessary to limit the number of routes per demand by considering some number of shortest routes only. Otherwise, the number of decision variables becomes too large to be solved. The objective (10.4) is to minimize the sum of the fixed ring costs and the incremental inter-ring routing costs. Constraint (10.5) ensures that for each demand bundle k, the sum of the flow on the routes $I(k)$ equals the total demand d_k. Constraint (10.6) ensures that on each span $s \in S(i)$ in route i, the sum of demand carried by the rings $J(s)$ that intersect span s is equal to the total flow F_i on route i. Constraint (10.7) states that the total demand G_{ij} carried on each span s of ring candidate $s \in J(s)$ does not exceed the aggregate capacity of all copies of ring candidate j. The total number of variables in this formulation is $|J| + |I| + \sum_{i \in I} |J(i)|$ and the total number of constraints is

$$|K| + \sum_{i \in I} |S(i)| + \sum_{j \in J} |S(j)| .$$

The main advantage of this formulation is that it jointly solves for the ring set and the routing pattern in a way that does *not* implicitly assume full span coverage. Therefore, it directly addresses the topology optimization (span elimination) problem. Although the formulation above models BLSR rings only, it can be extended to model UPSR rings with some modification. Similarly, decision variables for the location of ADMs in each candidate ring can also be added. These extensions are detailed in Appendix F but are not pursued here because of the large number of variables required to solve such problems. Even with the current formulation, the number of variables and constraints in moderately-sized problems can easily exceed the capability of modern optimization software. Nonetheless, this formulation has been used here to find feasible, but not provably optimal, solutions for networks with up to 32 nodes and 45 spans.

10.5 Foundation Design (FDIP)

Up to this point, we have defined a ring candidate in terms of its type, topological layout and capacity only, and have formulated the design problem so as to select a subset of these rings that serves all demands. In this last formulation, we extend our definition of a ring candidate to include a specification of the active and glass-through node locations and the subset of demands carried by each ring. Clearly, the number of all such candidates, even in relatively small networks, can be enormous. Therefore, the basic idea here is to restrict attention to a subset of *elite* candidates that have been frequently selected by other designs methods and are known to have high individual figures of merit by one or more different criteria (e.g., perfect capture, capacity utilization or cost/demand-served, etc.). In this situation, the problem is to select from this collection of elite candidates, a subset of rings that carries each demand at least once. In effect the logic of this approach is to present the combinatorial solver with an array of interesting or promising ring-type building blocks and simply let it handle the "final selection and assembly" aspect of arriving at a completed network design. The formulation for the *Foundation Design IP* (FDIP), is as follows:

FDIP

Minimize: $\sum_{\in J} c_j \cdot X_j$ (10.11)

Subject to: $\sum_{j \in J} \alpha_{jk} \cdot X_j \geq 1 ,$ $\forall k \in K$ (10.12)

$X_j \in \{0, 1\} ,$ $\forall j \in J$ (10.13)

The objective function (10.11) is to minimize the total design cost of fully specified rings selected from the elite ring set. Here the cost c_j is the actual cost of ring candidate j as specified by the location of active and glass-through nodes and the subset of demands carried by the ring. Constraint set (10.12) ensures that each demand is handled by at least one ring. Strictly speaking, this will not guarantee that all demands are served end-to-end but it is of research interest because the set of rings that are chosen may constitute a large, collectively optimized foundation upon which a complete design may be constructed. Moreover, the existing RingBuilder heuristic is ideally suited to identify a vocabulary of elite ring candidates. The total number of variables in this formulation is equal to the number of ring candidates and the total number of constraints is equal to the number of demands.

It is recognized apriori that the performance of FDIP will be the most difficult to assess comparatively because its aim is not even to produce a complete design and in producing a partial foundation design, its performance is the most dependent on the set of ring candidates presented for its consideration. Another way to put it is that FDIP will be highly dependent for its performance on both the pre-processing tactic to populate its vocabulary of elite ring candidates and on some second process to complete the resultant design working from the foundation FDIP contributed.

10.6 Comparative Study Method

To evaluate the performance of each formulation, we generate network designs for all twelve test cases outlined in Chapter 7 using the proposed formulations. All formulations are implemented in the AMPL mathematical programming language [Com97] and solved with the CPLEX linear optimization software [CPL96]. The AMPL models for these formulations are provided in Appendix G. The AMPL data for each problem instance are generated using several custom Perl, Python and Java programs. Each problem instance is solved using a parallel version of the CPLEX MIP Solver (Version 6.2) on a Sun UltraSparc HPC-450 equipped with 1 Gbytes of RAM and four processors, each running at 250 MHz. For comparative purposes, the runtime for all problem instances is limited to one hour of user-time. In those cases where the runtime is exceeded, the best feasible integer solution is reported. Note that the runtime for these cases varies because the actual CPU time consumed in one hour of user-time depends on the load on the computer. This is an unfortunate side-effect of using the parallel version of the CPLEX MIP. The variation in CPU time is not all that critical, however, because experience shows that doubling or tripling the runtime rarely improves the solution quality in such cases. Lastly, a tolerance on optimality was also set to

0.005. That is, problem instances that completed normally are guaranteed to be within 1/2 a percent from the optimal solution.

Because each formulation adopts a different objective or cost model, the objective values for each formulation are not directly comparable. Therefore, to calculate the cost of each solution for comparison, the ring sets were imported into RingBuilder Interactive and the demands were routed using the demand packing algorithm described in Chapter 8. This approach ensures that a uniform costing procedure is applied to all test cases. It also ensures independently that all demands are satisfied (i.e., routable) and the designs are feasible. The procedures for generating network designs using each formulation are discussed in detail below.

For the SCIP formulation, the working load on each span, w_s, is determined by routing the demand over the network topology using the shortest path algorithm described in Chapter 8. In all cases, route lengths are measured by geographical distance. The set of ring candidates J is generated using the candidate generation procedure described in Chapter 8. Due to the large number of nodes and spans in Net43 and its high average nodal degree, the entire cycle set is too large to enumerate completely. Therefore, a subset of cycles is enumerated by limiting the cycle circumference to 10 hops. The hop limit and the number of cycles for each test network are listed in Table 10.1. The cost of each candidate ring c_j is obtained by taking the sum of the fibre and ADM common equipment costs, assuming that an ADM is required at every node on the ring. After the solution (i.e., ring set) is imported into RingBuilder Interactive, the actual required number of ADMs, add-drop interfaces and inter-ring transitions are determined from the routing data.

Table 10.1: SCIP & FCRIP Design Parameters

	Net15		Net20		Net32		Net43	
Parameter	SCIP	FCRIP	SCIP	FCRIP	SCIP	FCRIP	SCIP	FCRIP
Hop Limit	15	8	20	10	32	18	10	6
Cycles	976	203	428	119	324	166	3,748	244

The same candidate set and (fixed) ring cost data are also used for the FCRIP data sets. The unit cost b_{ij} associated with carrying route i on ring j, is equal to the cost of two ADM add-drop interfaces and two DCS add-drop interfaces. This assumes that inter-ring transitions are cross-connected via a co-located DCS. The route set, I, is generated by finding the two shortest routes for each demand. The number of routes per demand is limited to restrict the number of variables in the integer program. In addition, the number of candidate rings is also reduced by limiting the cycle circumference (in hops). Even with these restrictions, the number of variables in the integer program can be enormous. For example, there are over 30,000 variables in the integer program for Net20 with a hop limit of 10 and only two routes per demand. The hop limit and the number of cycles for the FCRIP data set are listed in Table 10.1.

Given the large number of integer variables in the FCRIP data sets, the route flow F_i and ring flow G_{ij} variables were changed (or relaxed) to continuous variables to improve runtime. In practice, the relaxation of the flow variables is not a significant concern because they often take on integer values when solved. In addition, there is usually sufficient slack in the solution (i.e., ring set) to accommodate any rounding errors due to fractional flow values. Because the flow values may be fractional, however, the actual routing is determined by the demand packing algorithm.

To study the effect of the number of routes per demand in the FCRIP formulation on the total design cost, we also generate several network designs for Net32 by varying the number of routes per demand from 1 to 5.

For the FDIP formulation, the set of elite candidate rings and their respective route sets and costs for each test case are obtained from the network designs generated using the other design methods developed here. A Java program was written to extract the AMPL data from all network designs generated for each test case. The total number of designs and candidate rings for each test case is listed in Table 10.2.

Table 10.2: FDIP Data Set Statistics

Test Case	Network	Tech.	# Designs	# Candidate Rings
1	Net15	4B12	16	113
2	"	4B48	16	59
3	"	4B12, 4B48	16	77
4	Net20	4B12	16	175
5	"	4B48	16	82
6	"	4B12, 4B48	16	101
7	Net32	4B48	16	131
8	"	4B192	16	105

Table 10.2: FDIP Data Set Statistics

Test Case	Network	Tech.	# Designs	# Candidate Rings
9	"	4B48, 4B192	16	109
10	Net43	4B48	15	470
11	"	4B192	15	215
12	"	4B48, 4B192	15	241

In the FDIP formulation the cost, c_j, represents the cost of the actual ring in its parent design, as specified by the location of active and glassthrough nodes and the subset of demands served. Because the designs obtained using this method do not generally serve all demands, the foundation designs are completed by generating an incremental design for the remaining unserved demand using RingBuilder Interactive with the balanced ring loading algorithm.

10.7 Results

A summary of the total design cost and CPU time results for each formulation and test case are presented in Table 10.3. In some cases, the optimal solution to the IP formulations are not found by the CPLEX MIP Solver within the one-hour time limit. These instances are indicated by an asterisk in Table 10.3.

Table 10.3: Summary of Results

Test Case	Network	Tech.	SCIP		FCRIP		FDIP	
			Cost	Runtime (sec.)	Cost	Runtime (sec.)	Cost	Runtime (sec.)
1	Net15	4B12	67.45	71.3	55.35*	5,452	70.00	0.5
2	"	4B48	82.11	522.3	77.12*	6,433	63.62	0.02
3	"	4B12, 4B48	65.73	308.9	81.27*	6,471	55.77	0.19
4	Net20	4B12	120.36	0.53	105.79*	7,864	126.85	70.4
5	"	4B48	104.46	28.63	92.56*	8,333	111.01	0.10
6	"	4B12, 4B48	100.06	186.7	110.19*	6,402	99.46	0.16
7	Net32	4B48	876.72	0.10	912.46*	8,284	1,044.2	0.07
8	"	4B192	608.42	0.15	693.80*	8,642	928.9	0.01
9	"	4B48, 4B192	581.72	1.63	727.32*	8,265	847.6	0.02

Table 10.3: Summary of Results

			SCIP		FCRIP		FDIP	
Test Case	Network	Tech.	Cost	Runtime (sec.)	Cost	Runtime (sec.)	Cost	Runtime (sec.)
10	Net43	4B48	1,272.8*	4,508	-	-	1,488.8*	14,342
11	"	4B192	961.29*	6,464	-	-	967.53	7.15
12	"	4B48, 4B192	941.33*	5,424	-	-	945.93	9.52

* runtime limit exceeded before an optimal solution was found. Runtimes include total CPU time for all four processors.

A breakdown of the results for the initial IP solution and the final design for each formulation are also provided in Tables 10.5 through 10.7 at the end of this chapter. In these tables, the "objective" column lists the objective value for the best integer solution found by CPLEX within the runtime limit. The "lower bound" column gives a lower bound (from CPLEX) on the IP optimal solution and the "gap" is the percentage difference between the best integer solution and the lower bound. Also at the end of the chapter are the detailed statistics for all designs generated by these formulations.These results show that, despite a number of simplifying assumptions, the SCIP formulation provides the best solution (in terms of cost) in 8 out of 12 test cases. As a general outcome, however, this will be dependent on the relative cost/savings of a glassthrough node compared to typical line-distance costs in the solution because the SCIP formulation cannot resolve glassthrough locations. For example, in full metro contexts where there is zero line cost the solution could be much farther off the true minimum because getting the glassthroughs right may be "over half the battle." Table 10.5 shows that in all test cases, except those involving test network Net43, integer optimal solutions are found for the SCIP formulation within the runtime limit. In addition, in all test cases the final design satisfies all of the demands (i.e., all of the demands are packed or routed within the ring set). There is also a very high correlation (0.981) between the objective value of the IP solution and the total design cost. This means that even with the several simplifying assumptions, the objective function of the SCIP formulation provides a very good estimate of the total design cost. This result, however, is best suited to cases where the line costs dominate the total design cost.

The FCRIP formulation provides the best solution in the remaining four test cases. These include the single technology designs for test networks Net15 and Net20. The total design costs for these test cases are up to 18% lower than the corresponding SCIP designs. This demonstrates the potential savings from optimizing the ring set and demand routing simultaneously. Solutions are not available for test network Net43 because the data set for the problem instances cannot be generated, much less solved, within the available memory on the computer. And this is with a cycle circumference limit of only 6 hops!

As shown in Table 10.6, the runtime limit is exceeded in all test cases before an integer optimal solution is found. Furthermore, the "gap" between the objective value of the best integer solution and the lower bound increases with problem size (i.e., the number of variables). This is to be expected because it takes significantly longer to solve each LP relaxation in the CPLEX branch-and-bound tree as the problem size increases. As a result, fewer integer feasible solutions are explored within the runtime limit. This explains, in part, the inferior performance of the FCRIP formulation relative to the SCIP formulation on the larger problem instances (e.g. multi-technology designs). Another possible explanation for the relatively poor performance of the FCRIP method on larger problem instances is the low hop limit imposed on all designs to restrict the number of variables. This is supported by the observation that on Net20, the largest ring in the optimal SCIP solution has a circumference of only 8 hops; two less than the hop limit used for the FCRIP formulation. In all other test cases, however, the circumference of the largest ring in the SCIP solution is greater than the FCRIP hop limit.

The results in Table 10.6 also show that there is an extremely high correlation (0.9989) between the objective value of the IP solution and the total design cost. This is to be expected because the objective function for the FCRIP formulation models both the fixed ring costs and the inter-ring transition costs.

Note that in the final design for test case 4 (Net20, 4B12) only 98.1% of the demand can be packed into the ring set. An examination of the IP solution for this test case reveals that some of the flow variables take on fractional values. Therefore, there may not be a feasible (integer) demand routing for this test case. In practice, this may not be an issue because the demand matrix is typically based on forecasted data, which may vary substantially from the actual future demand. In all other test cases, the flow variables take on integer values and 100% of the demand is packed in the final design.

The effect of the number of routes per demand on the total design cost for test case 1 (Net15, 4B12) is shown in Figure 10.1. This figure also shows the integer objective value and the best bound from the IP solutions for each test case. The best bound is the objective value of the best unexplored node in the branch-and-bound tree for the problem, where each node represents an LP relaxation of the original problem. In all test cases, the runtime limit is exceeded before an optimal solution can be found.

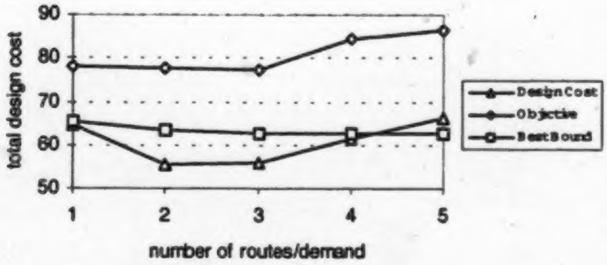


Figure 10.1. Effect of the number of routes/demand on FCRIP total design cost for test case 1 (Net15, 4B12).

These results show that although the best bound continues to decrease as the number of routes per demand increases, the lowest cost design occurs when there are only two routes per demand. This is because the number of decision variables increases dramatically as the number of routes per demand increases. For example, with only one route per demand there are 6,126 decision variables, whereas with five routes per demand there are 38,116 decision variables. Thus, a smaller portion of the solution space can be explored within the runtime limit when there are five routes per demand. This is also evidenced by the widening gap between the integer objective and the best bound at larger parameter values.

As shown in Table 10.3, the solution times for the FDIP formulation are generally an order of magnitude lower than the other formulations. Note that these times do not include the preprocessing required to generate the elite ring candidate set, which can be significant. In three test cases, the FDIP formulation also provides the lowest cost solution but in some cases the total design cost (including the incremental systems) is significantly higher than the other two methods. One possible cause for the high design costs becomes evident upon examination of the foundation designs for those test cases involving Net32. In this network, all demands terminate at one of three hub nodes. As a result, the rings in the foundation design are clustered about the three hub nodes, leaving a large portion of the network uncovered. Consequently, when the design is imported into RingBuilder Interactive and the demand is packed, only a small fraction of the demands are served from end-to-end. For example, in test case 8 (4B48) only 15.5% of the demand is actually served by the foundation design. Thus, the final design is generated primarily by the greedy heuristic algorithm (with demand packing). As a result, the correlation between the objective values from the IP solution and the final network designs is much lower than the other two formulations (i.e., 0.905).

One possible approach for improving performance is to reformulate the problem to include some measure of the degree to which a demand is satisfied from end-to-end. For example, if the variable α_{jk} represented the fraction of the distance that demand k is carried from end-to-end along ring j, then constraint (10.12) may provide a better indication of the demand served. Another tactic would be to considered only those demands in the candidate rings that are fully captured by the ring, i.e., that originate and terminate on the ring. These and other tactics along with the effect of the size and composition of the elite candidate ring set remain topics for future study.

10.8 Summary

We have discussed three new or recent approaches that are based on formal mathematical programming methods, for the multi-ring design problem. A summary of their respective features and advantages is given in Table 10.4.

Table 10.4: Comparative Summary of IP Formulations

Model Attribute	SCIP	FCRIP	FDIP
Glass-through	N	N	Y
Multi-modularity	Y	Y	Y
Span Elimination	N	Y	Y
Inter-ring transitions	N	N	Y
Ring Types	BLSR	BLSR/UPSR	BLSR/UPSR
Relative Run Times	Moderate	Slow	Fast
Design Completeness	Complete	Complete	Incomplete
Pre-processing	Minimal	Moderate	Greatest

The results indicate that these approaches can provide useful solutions to a difficult optimization problem. Like many approaches based on integer programming, however, there are inherent limitations in the scalability of these techniques to larger network designs, as illustrated by the results presented here. Nonetheless, these formulations can generate good solutions for relatively small networks and serve as useful benchmarks for other methods. Future work in this area may involve characterizing the various trade-offs between solution quality, candidate set size and composition, and model fidelity as well as exploring advanced combinatorial optimization techniques such as column generation.

Table 10.5: Computational Results for SCIP Formulation

Test Case	Network	Tech.	Objective	Lower Bound	Gap (%)	Runtime (sec.)	Demand Served (%)	Total Design Cost
1	Net15	4B12	70.606	70.323	0.4	71.3	100	67.452
2	"	4B48	53.852	53.626	0.4	522.3	100	82.106
3	"	4B12, 4B48	49.032	48.789	0.5	308.9	100	65.732
4	Net20	4B12	100.264	100.120	0.1	0.53	100	120.36
5	"	4B48	84.964	84.545	0.5	28.63	100	104.46
6	"	4B12, 4B48	73.550	73.187	0.5	186.7	100	100.06
7	Net32	4B48	790.22	790.22	0.0	0.10	100	876.72
8	"	4B192	525.82	523.55	0.4	0.15	100	608.42
9	"	4B48, 4B192	493.82	493.82	0.0	1.63	100	581.72
10	Net43	4B48	913.46*	890.12	2.6	4,508	100	1,272.8*
11	"	4B192	621.84*	557.86	11.5	6,464	100	961.29*
12	"	4B48, 4B192	611.18*	483.19	26.5	5,424	100	941.33*

*runtime limit exceeded before an optimal solution was found. Runtimes include total CPU time for all four processors.

Table 10.6: Computational Results for FCRIP Formulation

Test Case	Network	Tech.	Objective	Lower Bound	Gap (%)	Runtime (sec.)	Demand Served (%)	Total Design Cost
1	Net15	4B12	77.650*	63.516	22.3	5,452	100	55.350*
2	"	4B48	98.919*	46.969	110.6	6,433	100	77.119*
3	"	4B12, 4B48	105.168*	46.969	123.9	6,471	100	81.268*
4	Net20	4B12	142.14*	131.919	7.8	7,864	98.9	105.79*
5	"	4B48	127.562*	92.978	37.2	8,333	100	92.562*
6	"	4B12, 4B48	148.686*	92.372	61.0	6,402	100	110.19*
7	Net32	4B48	908.210*	631.734	43.8	8,284	100	912.46*
8	"	4B192	721.300*	252.257	185.9	8,642	100	693.80*
9	"	4B48, 4B192	784.020*	252.071	211.0	8,265	100	727.32*
10	Net43	4B48	-	-	-	-		-
11	"	4B192	-	-	-	-		-
12	"	4B48, 4B192	-	-	-	-		-

*runtime limit exceeded before an optimal solution was found. Runtimes include total CPU time for all four processors.

Table 10.7: Computational Results for FDIP Formulation

Test Case	Network	Tech.	Objective	Lower Bound	Gap (%)	Runtime (sec.)	Demand Served (%)	Total Design Cost
1	Net15	4B12	46.794	46.794	0.0	0.5	73.3	70.00
2	"	4B48	50.437	50.437	0.0	0.02	96.6	63.62
3	"	4B12, 4B48	45.01	45.01	0.0	0.19	90.29	55.77
4	Net20	4B12	91.875	91.432	4.9	70.4	73.3	126.85
5	"	4B48	80.480	80.480	0.0	0.10	95.1	111.01
6	"	4B12, 4B48	73.950	73.950	0.0	0.16	80.2	99.46
7	Net32	4B48	452.78	451.17	0.4	0.07	51.7	1,044.2
8	"	4B192	336.09	336.09	0.0	0.01	15.5	928.9
9	"	4B48, 4B192	279.13	279.13	0.0	0.02	14.1	847.6
10	Net43	4B48	1,112.36	847.58	31.2	14,342	84.2	1,488.83*
11	"	4B192	797.28	796.00	1.6	7.15	97.7	967.53
12	"	4B48, 4B192	763.21	763.21	0.0	9.52	95.1	945.93

*runtime limit exceeded before an optimal solution was found. Runtimes include total CPU time for all four processors.

Table 10.8: Design Statistics for SCIP Formulation

Test Case	Net-work	Number of Rings			Number of ADMs			Regen.	Fibre (km)	Avg. Fill (%)	Inter-ring Transit.	Avg. Hops/ Route	Spans Elim.	Total Working (DS3-hops)	Total Capacity (DS3-hops)	Total Cost	Runtime (sec.)
		4B12	4B48	4B192	4B12	4B48	4B192										
1	Net15	9	-	-	50	-	-	0	770.5	65.0	33	1.89	0	390	600	67.452	71.3
2	"	-	5	-	-	33	-	1	521.3	28.1	30	2.22	0	458	1,632	82.106	522.3
3	"	5	2	-	30	8	-	0	606.4	57.5	64	2.08	0	428	744	65.732	308.9
4	Net20	18	-	-	81	-	-	8	2,252	68.7	91	2.11	0	734	1,068	120.36	0.53
5	"	-	7	-	-	38	-	2	992.8	45.2	58	2.49	0	867	1,920	104.46	28.63
6	"	4	5	-	17	24	-	3	1,110	51.1	120	2.04	0	711	1,392	100.06	186.7
7	Net32	-	12	-	-	75	-	353	122,444	45.7	262	5.51	1	1,951	4,272	876.72	0.10
8	"	-	-	7	-	-	45	190	67,564	21.8	349	5.55	3	1,966	9,024	608.42	0.15
9	"	-	4	4	-	13	30	192	67,564	28.7	418	5.45	3	1,929	6,720	581.72	1.63
10	Net43	-	37	-	-	243	-	144	84,292	79.7	2,103	3.73	4	9,413	11,808	1,272.8*	4,508
11	"	-	-	15	-	-	109	62	37,168	48.3	2,008	4.01	4	10,118	20,928	961.29*	6,464
12	"	-	8	13	-	24	87	75	40,236	55.9	2,029	4.03	4	10,178	18,192	941.33*	5,424

*runtime limit exceeded before an optimal solution was found. Runtimes include total CPU time for all four processors.

Table 10.9: Design Statistics for FCRIP Formulation

Test Case	Net-work	Number of Rings			Number of ADMs			Regen.	Fibre (km)	Avg. Fill (%)	Inter-ring Transit.	Avg. Hops/ Route	Spans Elim.	Total Working (DS3-hops)	Total Capacity (DS3-hops)	Total Cost	Runtime (sec.)
		4B12	4B48	4B192	4B12	4B48	4B192										
1	Net15	6	-	-	39	-	-	1	689.9	83.8	24	1.95	3	402	480	55.350*	5,452
2	"	-	5	-	-	31	-	0	483.8	26.0	24	1.88	3	387	1,488	77.119*	6,433
3	"	0	5	-	0	33	-	1	553.6	23.1	20	1.83	4	377	1,632	81.268*	6,471
4	Net20	14	-	-	69	-	-	3	1,858	82.2	97	2.04	0	710	864	105.79*	7,864
5	"	-	6	-	-	30	-	0	772.4	50.2	113	2.08	6	723	1,440	92.562*	8,333
6	"	0	8	-	0	39	-	2	977.2	37.3	95	2.11	2	734	1,968	110.19*	6,402
7	Net32	-	12	-	-	69	-	381	129,572	45.3	327	5.41	0	1,915	4,224	912.46*	8,284
8	"	-	-	8	-	-	48	238	82,460	17.7	242	5.39	0	1,907	10,752	693.80*	8,642
9	"	-	2	6	-	7	37	276	92,304	19.7	309	5.89	1	2,084	10,560	727.32*	8,265
10	Net43	-		-	-		-										
11	"	-	-		-	-											
12	"	-			-												

*runtime limit exceeded before an optimal solution was found. Runtimes include total CPU time for all four processors.

Table 10.10: Design Statistics for FDIP Formulation

Test Case	Net-work	Number of Rings			Number of ADMs			Regen.	Fibre (km)	Avg. Fill (%)	Inter-ring Transit.	Avg. Hops/ Route	Spans Elim.	Total Working (DS3-hops)	Total Capacity (DS3-hops)	Total Cost	Runtime (sec.)
		4B12	4B48	4B192	4B12	4B48	4B192										
1	Net15	9	-	-	49	-	-	8	960.3	63.7	43	2.12	2	436	684	70.00	0.5
2	"	-	4	-	-	24	-	9	484.0	30.0	11	2.31	4	475	1,584	63.62	0.02
3	"	4	1	-	18	11	-	8	514.3	66.4	13	2.71	6	558	840	55.77	0.19
4	Net20	12	-	-	85	-	-	32	2,810	80.3	40	3.24	2	1,127	1,404	126.85	70.4
5	"	-	6	-	-	41	-	9	1,262	46.1	35	3.18	3	1,107	2,400	111.01	0.10
6	"	3	3	-	8	31	-	11	1,253	64.7	36	3.73	4	1,297	2,004	99.46	0.16
7	Net32	-	16	-	-	87	-	430	147,364	41.5	297	6.19	2	2,192	5,280	1,044.2	0.07
8	"	-	-	10	-	-	63	333	113,528	12.8	250	5.64	2	1,996	15,552	928.9	0.01
9	"	-	7	4	-	35	29	323	110,860	24.6	250	6.24	1	2,210	8,976	847.6	0.02
10	Net43	-	40	-	-	286	-	230	114,016	61.4	1,745	3.70	4	9,341	15,216	1,488.8*	14,342
11	"	-	-	14	-	-	102	105	45,696	41.7	1,838	3.93	17	9,922	23,808	967.53	7.15
12	"	-	8	7	-	52	67	99	48,476	67.4	1,855	4.26	14	10,748	15,936	945.93	9.52

*runtime limit exceeded before an optimal solution was found. Runtimes include total CPU time for all four processors.

11 A Tabu Search Meta-Heuristic for Multi-Ring Network Design

11.1 Introduction

In this chapter, we develop a novel Tabu Search meta-heuristic for the multi-ring network design problem. *Tabu Search* (TS) is a meta-heuristic that guides a local neighbourhood search to explore regions in the solution space beyond a local optimum [GIL97]. A *local neighbourhood search* begins with an initial (sub-optimal) solution for a given problem and searches its immediate neighbourhood for a better solution. A solution's *neighbourhood* is defined as the set of adjacent solutions that can be reached directly by an operation called a *move*. Generally, a move involves changing some of the attributes of the current solution. In a Knapsack Problem, for example, a typical move is to swap an element already in the solution with one that is not. Aside from Tabu considerations (to follow), when a better solution is found, the current best solution is replaced and the search begins again from the better solution. This process continues until no improvement can be made to the current solution. This type of search is sometimes called a *steepest-descent search*. Although the final solution is (locally) optimal with respect to its immediate neighbourhood, it is unlikely to be a globally optimal solution unless the solution space is convex. One approach for dealing with local optimality is to accept "uphill moves" to an inferior solution after reaching a local optimum. Using this approach, the search can escape local minima and go on to explore other promising regions in the solution space. This is the approach adopted in TS. A graphical illustration of this basic idea is shown in Fig. 11.1. Note that in combinatorial problems the solution space consists of a large but finite set of discrete points in n-dimensional space and is not continuous as suggested by Fig. 11.1.

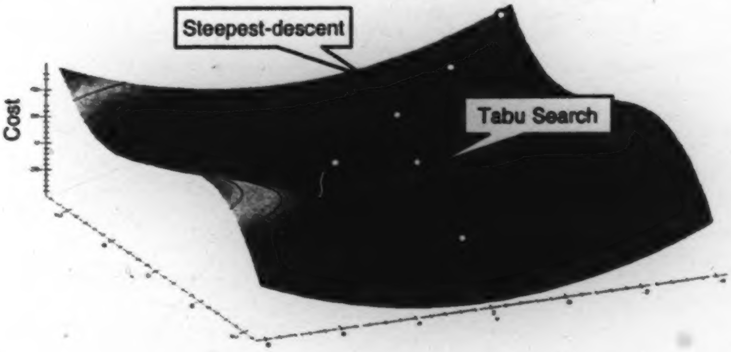


Figure 11.1. A graphical illustration of the Tabu Search meta-heuristic.

Clearly, some restrictions need to be imposed on accepting uphill moves, otherwise the procedure may search the entire solution space. In TS, data structures are used to control the search process and the acceptance of uphill moves. These structures maintain a history of the states encountered during the search and are used to determine the solutions that may be reached by a move from the current solution. Using these structures, TS attempts to imitate an "intelligent" search process [GIL97]. This differs from similar search procedures such as Monte Carlo search and Simulated Annealing, which use a random process to select a neighboring solution and as a means of leaving local optima [Ree93].

The basic TS procedure is quite straight forward. The important aspects of the method are the way in which the history is kept, defined and updated and how the evaluation function is determined. In its simplest form, memory is used to record key attributes of prior moves to prevent the search from revisiting previous solutions (i.e., cycling). The main mechanism for preventing cycling is to restrict the neighbourhood of the current solution by forbidding (or penalizing) certain *tabu* moves. Once a move is marked as tabu, it usually remains in that state for a specified number of subsequent moves. This exclusion period is called the *tabu tenure* of a move.

Let x_i be the solution at the current iteration i, x_i^* be the best solution found so far, and T_i be the set of tabu moves at iteration i. A basic version of TS may be expressed by the pseudo-code in Fig. 11.2.

```
Step 1: $i = 0$; initialize $x_0$; $x_0^* = x_0$; $T_0 = \varnothing$.
Step 2: Construct a list of candidate moves from the neighbourhood of $x_i$. Evaluate each candidate move.
Step 3: Select the highest ranked move that doesn't belong to $T_i$, or is admissible by aspiration. Perform the move and update $x_i$.
Step 4: If $x_i$ is better than $x_i^*$, update $x_i^*$.
Step 5: If the stopping criteria are satisfied, terminate with $x_i^*$. Otherwise, $i = i + 1$; update $T_i$; go to Step 2.
```

Figure 11.2. Basic version of Tabu Search meta-heuristic (adapted from [GIL97]).

A common extension of the basic TS procedure is to allow a move that is marked as tabu if it leads to a highly desirable solution. Such a move is said to be *admissible by aspiration*. Another enhancement is to add a diversification procedure that restarts the search in a previously unexplored region of the solution space whenever the rate of improvement drops sufficiently or a pevious solution is revisited. Several other advanced strategies have been proposed for enhancing this basic form of TS [GIL97], [Bat96], [Ree93]. TS has been applied to a variety of combinatorial optimization problems with considerable success. It is this success that motivated us to develop and test a TS design method for the multi-ring network design problem.

The remainder of this chapter is organized as follows. In Sections 11.2 and 11.3 we describe the basic neighbourhood and memory structures used in the TS procedure developed here for the multi-ring network design problem. This is followed by a description of the overall search procedure in Section 11.4 and the search diversification strategy in Section 11.5. In Section 11.6, we describe the study method used to evaluate the effect of TS parameters on the solution value. The experimental results are given in Section 11.7 followed by some concluding remarks in Section 11.8.

11.2. Neighbourhood Structure

For the purposes of the search procedure, we define a solution as a set of rings only and ignore the routing pattern. This is similar to, and finds some justification as part of a practical design strategy, from the concept of implicit routing in the widely used MENTOR algorithm for private-line network design [KKG91]. For a given solution to be feasible, it must be able to serve all demands. The rings in the solution are selected from a set of candidate rings, which are generated using the procedure described in Chapter 8. The moves that define the local neighbourhood of a solution are:

(1) *constructive move*: copy a ring from the set of candidate rings and add it to the solution. This increases the number of rings in the solution by one.

(2) *destructive move*: remove a ring from the current solution. This decreases the number of rings in the solution by one.

Note that this definition of the local neighbourhood may include solutions that are not feasible. That is, it may not be possible to serve all of the demand if a ring is dropped from the solution. All feasible solutions in the solution space, however, can be reached by a sequence of constructive and destructive moves.

11.3 Memory Structures

Tabu memory structures play an important role in the process that guides the local search process. Two types of memory are used here to control the search process: short-term and long-term. Short-term memory is used to prevent the search from being trapped in a local optimum, while long-term memory is used to diversify the search process.

The short-term memory imposes restrictions to discourage the reversal of recently made moves. For example, if ring candidate r1 is added to the current solution, we discourage it from being dropped from the solution for a specified number of iterations, called the *tabu drop tenure*. Similarly, if a ring r2 is dropped from the solution, we discourage it from being added back to the solution for the *tabu add tenure*. These restrictions are imposed by penalizing the evaluation metric of moves that are in the tabu state. This simply discourages the search procedure from making tabu moves rather than rejecting them outright, which could prevent a feasible solution from being constructed for example. For constructive moves, we use the transport efficiency of the network design as a whole (i.e., the total demand served divided by the total cost) as the evaluation metric. We refer to this as the *global transport efficiency*. The global transport efficiency is chosen for the constructive moves because it is easily calculated and gives a better indication of the total contribution that a candidate ring makes to a design than the individual transport efficiency. In contrast, we use the individual ring's transport efficiency as the evaluation metric for destructive moves because the demand packing algorithm would need to be called to properly calculate the global transport efficiency. The candidate ring that provides the highest global transport efficiency is selected for constructive moves and the ring with the lowest transport efficiency is selected for destructive moves at each iteration.

In the current algorithm, separate tabu tenure parameters are provided for both types of move. In general, the tabu tenure of destructive (drop) moves is set to a smaller value than the tabu tenure for constructive (add) moves. This is because the set of candidate rings is usually much larger than the number of rings in the solution. Therefore, using a single tabu tenure would be more restrictive for drop moves than add moves. In addition we know from problem understanding that the desired lower design cost usually lies in the direction of fewer rings in total, not towards a net addition of rings from the starting design. The longer the tabu tenure, the less likely the search will revisit a previous solution. Conversely, the shorter the tabu tenure the more likely it is that the search will converge on the local optimum quickly. Therefore, a proper balance is required to achieve the best performance.

We implement the short-term memory by storing the iteration number at which the add and drop moves for each ring candidate are no longer tabu. Let *tabu_add(r)* and *tabu_drop(r)* be the iteration number at which the tabu status for add and drop moves, respectively, expires for candidate ring *r*. Let *i* be the current iteration number and let *add_tenure* and *drop_tenure* be the parameters for the add and drop moves, respectively. Initially, the *tabu_add(r)* and *tabu_drop(r)* values are set to zero for all candidate rings. Each time an add move r is made we update the *tabu_drop(r)* variable as follows:

$$tabu\_drop(r) = i + drop\_tenure \quad (11.1)$$

Similarly, each time an drop move r is made we update the *tabu_add(r)* variable as follows:

$$tabu\_add(r) = i + add\_tenure \quad (11.2)$$

Therefore, a move that involves dropping candidate ring r from the solution is *tabu* if and only if $i < tabu\_drop(r)$. Similarly, a move that involves adding candidate ring r is *tabu* if and only if $i < tabu\_add(r)$. Rather than rejecting tabu moves outright, we use a penalty structure to discourage tabu moves from being made. Add moves that are in the tabu state are penalized by dividing their evaluation metric by a user-defined *penalty factor*, whereas drop moves are penalized by multiplying their evaluation metric by a penalty factor. Separate penalty factors are provided for both types of move. Penalties are used in favour of outright rejection because there can be situations in which there is no other alternative but to make a tabu move. This can occur, for example, when all of the drop moves are in the tabu state (e.g., when the number of rings is less than the drop tenure). It can also occur during an add move when only one candidate ring is able to serve a selected set of demands (e.g., for topological reasons). The penalty on tabu moves is also temporarily lifted if the move leads to a new best solution. That is, the evaluation metric of either move is not divided or multiplied by a penalty factor when a new best solution is found. This is called an *aspiration criteria*.

In addition to short-term memory, we also use long-term memory to implement a search diversification strategy that encourages the search to explore unvisited regions of the solution space. This is done by generating a new starting solution based on information contained in all previous solutions. In doing so, it is desirable to choose a set of rings for the new solution that is significantly different from those found in all previous solutions. This is to avoid being trapped in the same *attraction basin* as the previous solutions.

In general, there are several factors that can be used to guide the selection of a new ring set (i.e., solution). Two of these are the frequency with which a candidate ring has appeared in all previous solutions and its relative influence on those solutions. Here, we use the cumulative demand served by each candidate ring by each candidate ring as a combined measure of both its frequency and influence. That is, whenever a new solution is found we accumulate the amount of demand served by each candidate ring in long-term memory.

Candidate rings that exhibit a relatively high degree of influence and appear frequently in the previous solutions are likely candidates that should be eliminated from the new starting solution.

Candidate rings that have a relatively low degree of influence, but appear frequently in previous solutions, are usually good "crack-fillers."

11.4 Search Procedure

A flow chart of the overall TS procedure is shown in Figure 11.3. This procedure is based on an advanced TS strategy known as Strategic Oscillation [GIL97]. The basic idea behind this procedure is to alternate between the feasible and infeasible regions of the solution space by making elementary add and drop moves.

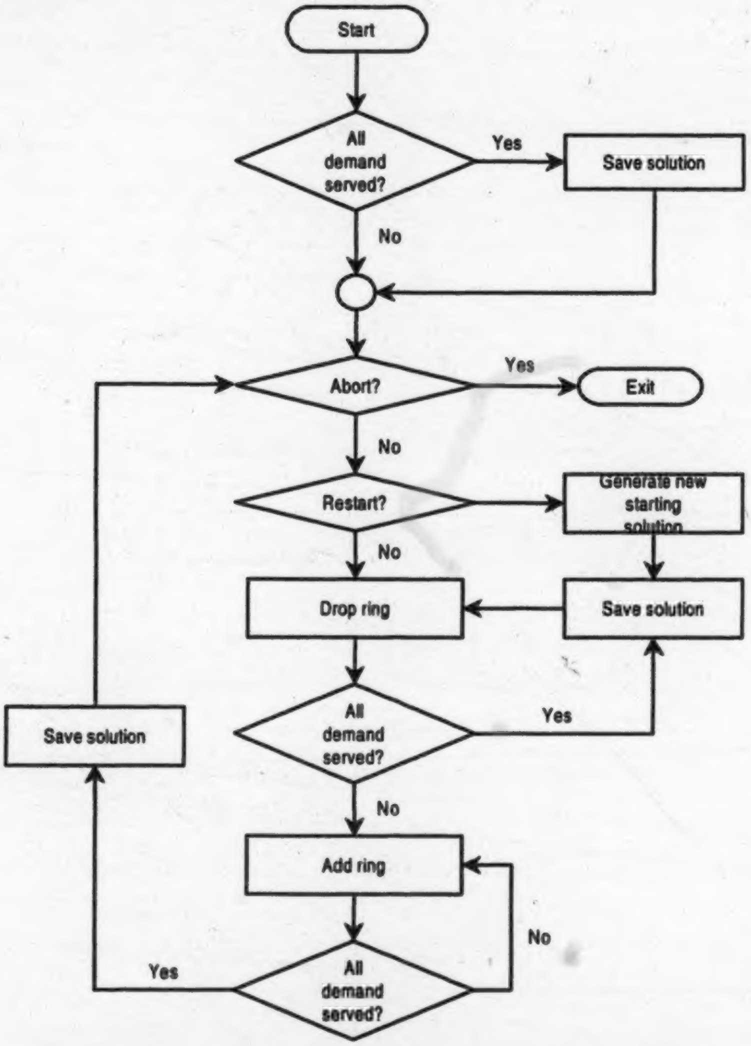

Start
All demand served?
Yes
Save solution
No
Abort?
Yes
Exit
No
Restart?
Generate new starting solution
No
Drop ring
Save solution
All demand served?
Yes
Save solution
No
Add ring
No
Yes
All demand served?

Figure 11.3. Flow chart of basic Tabu Search meta-heuristic.

The procedure is broken down into four main phases: initialization, a destructive phase, a constructive phase and search diversification. In the first phase of the procedure, an initial solution is constructed using the RingBuilder algorithm described in Chapter 8. Note that the initial design can be generated using any ring design method but for simplicity the RingBuilder algorithm was used here. If the RingBuilder algorithm is not used to create the initial solution, a set of candidate rings is generated using the same procedure as described in Chapter 8. Otherwise, the set of candidate rings generated by the RingBuilder algorithm is reused. Each candidate ring is specified by its type, capacity and topological layout. This set of candidates is the available pool of building blocks from which new solutions are constructed. If the initial solution serves all demand, it is saved as the current best solution and a hash code for the solution is computed and saved in a list. This list is used to detect when a previous solution has been revisited. A hash code is used here rather than a complete specification of the solution to conserve memory. The hash code is generated by taking the sum of the hash codes of each ring in the solution. The hash code for each ring is derived from Object.hashCode() method in the Java programming language.

Next, the procedure enters the Destructive Phase. In this phase, the transport efficiency of the rings in the current solution is calculated and the ring with the lowest utilization (after applying any tabu penalties) is dropped from the current solution. The demands formerly served by this ring are then packed (to the extent possible) into the slack in the remaining rings. A modified version of the Demand Packing algorithm described in Chapter 8 is used here to pack unserved demands. The main difference is that the version used here removes all routes served by the deleted ring before beginning the packing process. This is possible because it is not necessary in the current algorithm to reconstruct any routes. Therefore, removing the routes prior to packing increases the likelihood of successfully packing all demands. If all the demand are packed, the cost of the current (feasible) solution is calculated. If the cost of the current solution is lower than the previous best solution, the previous best solution is replaced with the current solution. This process continues until the fraction of demand served from end-to-end drops below a specified threshold, called the *drop depth*.

At this point, the search procedure enters the Constructive Phase. At each iteration in this phase, the candidate rings are evaluated by temporarily adding them to the solution, calling the demand packing algorithm and calculating the overall transport efficiency based on the total demand served and the total ring cost. If the move is in the tabu state, the transport efficiency is divided by a penalty factor. The candidate ring that yields the highest overall efficiency is then selected and added to the solution. This process continues until a feasible solution is found (i.e., all of the demand is served). At this point, the cost of the current solution is calculated and the best solution is updated if required. It is worth noting here that while the solution is in the infeasible region the measure of transport efficiency applies, but any time the solution enters feasible region it is the true design cost that is used to measure solution quality.

The procedure then returns to the Destructive Phase and the entire process continues until either the search diversification or stopping conditions are met. Search diversification is initiated when either a previous solution has been revisited or the rate of improvement in the solution cost drops below a preferred level. Two ways are used to measure the rate of improvement: relative gap and absolute gap. The *relative gap* is the relative change in the best solution cost over a given number of prior iterations, known as the *restart window*. The *absolute gap* is the absolute change in the best solution cost over some number of prior iterations. For example, if the best solution cost at the beginning and end of the restarting window were 100 and 95, respectively, the relative gap would be $(100-95)/100 = 0.05$ and the absolute gap would be $100-95 = 5$. The search diversification strategy is described in detail in the next section. The stopping conditions include limits on the number of drop/add iterations, memory usage, runtime and the relative or absolute rate of improvement in the best solution cost.

11.5 Search Diversification Strategy

Whenever the diversification conditions are met, the RingBuilder heuristic is used to generate a new starting solution. However, the candidate ring evaluation process is biased during the synthesis process to penalize candidate rings according to their influence on all previous solutions. The biased transport efficiency η_j' for ring *j* is given by the following expression

$$\eta_j' = \eta_j \cdot \left(1 + p \cdot v_j / \sum_{i=1}^{n} v_i\right)^{-1} \tag{11.3}$$

where η_j is the transport efficiency of candidate ring *j*, *p* is a constant restart penalty factor, v_j is the cumulative (distance-weighted) demand served by candidate ring *j* in all prior solutions and *n* is the number of candidates rings. The term $v_j/\sum_{i=1}^{n} v_i$ is the fraction of demand served by candidate ring *j* in all previous solutions. This provides a relative measure of a candidate ring's influence on past events. In general, the candidate rings with the highest influence will have occurred more frequently in past solutions and served a large portion of the total demand within those solutions. In some cases, they may have been part of the initial starting solution and were never dropped in subsequent solutions. Such candidate rings are best avoided when generating a new starting solution to help ensure that the search procedure restarts in an unexplored region of the solution space.

Otherwise, the search may converge to the same solutions as seen previously. The biased transport efficiency in Eq. (11.3) achieves the desired behaviour by reducing the transport efficiency of candidate rings by their relative influence on previous solutions.

11.6 Comparative Study Method

To assess the performance of TS we generate network designs for all twelve test cases in Chapter 7. The initial starting solution for each test case is obtained from the RingBuilder heuristic with balanced ring loading and no demand packing. For each test case, the number of iterations is limited to 100 and the runtime is limited to one hour. The default tabu parameters listed (in boldface) in Table 11.1 are used in all test cases. The absolute restart gap and the relative and absolute stop gap are set to infinity.

To evaluate the effect of various parameter settings on the total design cost, we generated a series of network designs for test network Net20. Here, the initial starting solution for each test case is obtained from the RingBuilder heuristic with unbalanced ring loading and no demand packing. In all cases, we limited ourselves to single technology designs consisting of 4 fibre, OC-48 BLSR (4B48) rings. In total, 55 network designs are generated for a variety of parameter setting. A list of the parameters and the range over which they are tested is shown in Table 11.1.

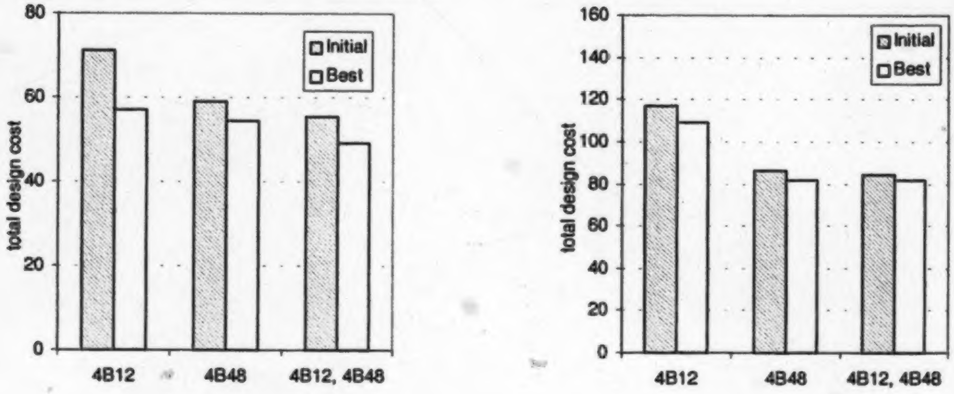
Table 11.1: Test Parameters Settings (default values in boldface)

Parameter	Test Values
Add Tenure	2, **3**, 4, 6, 8, 10
Drop Tenure	2, **3**, 4, 6, 8, 10
Add Penalty	2, 4, **8**, 16
Drop Penalty	2, 4, **8**, 16
Drop Depth	0.6, 0.8, 0.7, 0.9, **1.0**
Restart Window	5, **10**, 15, 20, 25
Restart Gap	**0.05**
Restart Penalty	2, 5, **10**, 15, 25
ADM Discount Factor	0.0, 0.05, **0.1**, 0.15, 0.2

To limit the number of test cases, each parameter is tested over the specified range while holding the other parameters at their default values (in boldface). The only exceptions to this rule are the add and drop tenures. Here, all combinations of the specified Add and Drop Tenures are evaluated. This constitutes the majority of the test cases. In each case, the number of iterations is limited to 100 and the runtime is limited to one hour. The Absolute Restart Gap and the Stop Gap and Absolute Stop Gap were set to infinity.

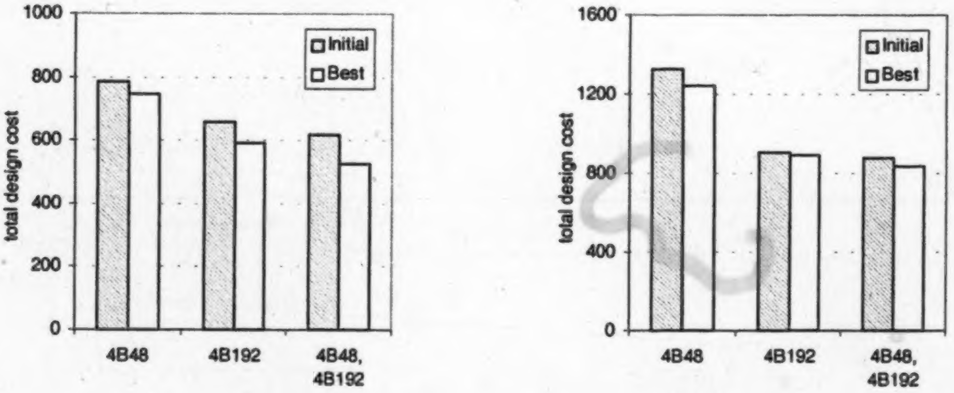
11.7 Results

A summary of the results is provided in Fig. 11.4. Detailed statistics for each design generated with the TS meta-heuristic are provided in Table 11.3 at the end of this chapter.



(a) Net15

(b) Net20



(c) Net32

(d) Net43

Figure 11.4. Results for Tabu Search meta-heuristic.

The results show that the TS meta-heuristic generates designs that are up to 20% lower than the initial starting solution. On average, the TS meta-heuristic provides a 8.1% reduction in total design cost relative to the initial starting solution. Average runtime per iteration, however, increases dramatically for the larger problems, i.e. those involving Net43. For example, the average runtime per iteration (i.e., solution) for test case 1 (Net15, 4B12) is 24.1 seconds, whereas the average runtime per solution for test case 9 (Net43, 4B192) is 393 seconds. As a result, test case 9 only generates 39 solutions within the one hour time limit. Nonetheless, the TS meta-heuristic is still significantly faster than the dithered sequencing approach, for example, which requires over 6,500 seconds to generate each solution.

A typical example of the progress of the TS algorithm for test case 5 (Net20, 4B48) is shown in Figure 11.5. This figure plots the total design cost of the current and best solution as a function of the iteration number. This plot demonstrates the ability of the TS meta-heuristic to escape local minima in the solution space. For example, in Fig. 11.5 the search proceeds through several local minima and is restarted twice before finding its best solution at iteration 41. In total, the search is restarted seven times in one hundred iterations, once because a previous solution is revisited and the remaining times because the restart gap is exceeded. Each time the search is restarted (i.e., a new starting solution is generated), the total design cost returns to roughly the same level as the initial starting solution. This is not surprising because the RingBuilder algorithm is used to create both the initial and the new starting solutions. At each subsequent iteration, the cost decreases gradually at first as small, lightly loaded rings are removed from the solution and then more abruptly as larger, more influential rings are removed. Eventually, the demand served drops below the drop depth threshold and a ring is added to the solution to restore feasibility, causing the total design cost to rise. In this test case, the search procedure alternates between the destructive (drop) and constructive (add) phases a few times before the restarting gap is reached.

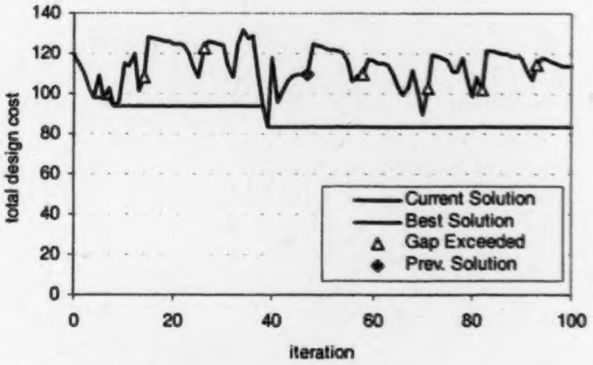


Figure 11.5. Example of TS search trajectory.

The network designs for the initial starting solution and the TS solution for test case 5 (Net20, 4B48) are shown in Figs. 11.6 and 11.7, respectively. In these figures ADMs are represented by small squares overlaid on each ring. The initial starting solution, generated by the RingBuilder algorithm, contains a total of four rings with a total fibre mileage of 1,066 km. The rings are numbered in the order that they were selected. There are a total of 29 ADMs, 11 glassthroughs and 35 inter-ring transitions in the initial design. Therefore, on average each ring is 10 hops in length and contains 7.25 ADMs and 2.75 glassthroughs. Note that the rings that are selected last generally have fewer hops than those selected in earlier iterations. This is because there are fewer fragments of demand to be served as the design progresses and these fragments are usually best served by rings with the least number of hops.

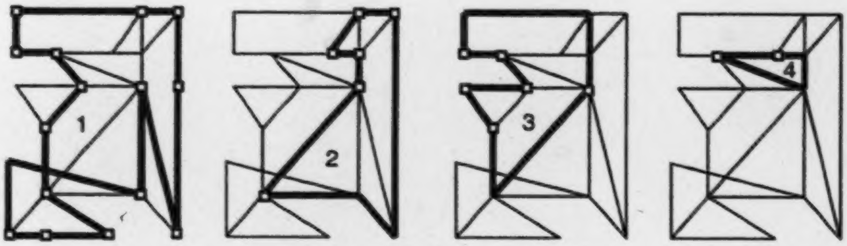


Figure 11.6. Initial starting solution for Net20.

In contrast, the Tabu Search design (evolved from this starting point) contains three rings with a total fibre mileage of 794 km (26% lower than the initial design). The TS design has a total of 28 ADMs, 2 glassthroughs and 39 inter-ring transitions. The rings in this case also have an average length of 10 hops but there are significantly fewer glassthroughs. As a result, there are a greater number of ADMs per ring on average. Although there are fewer rings in the TS design, only four more inter-ring transitions are required.

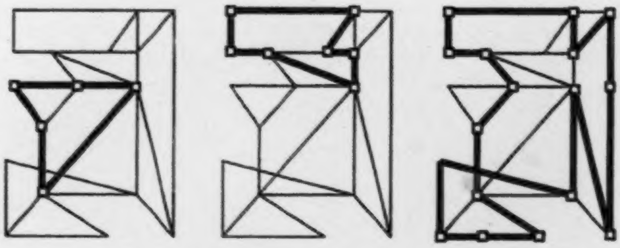
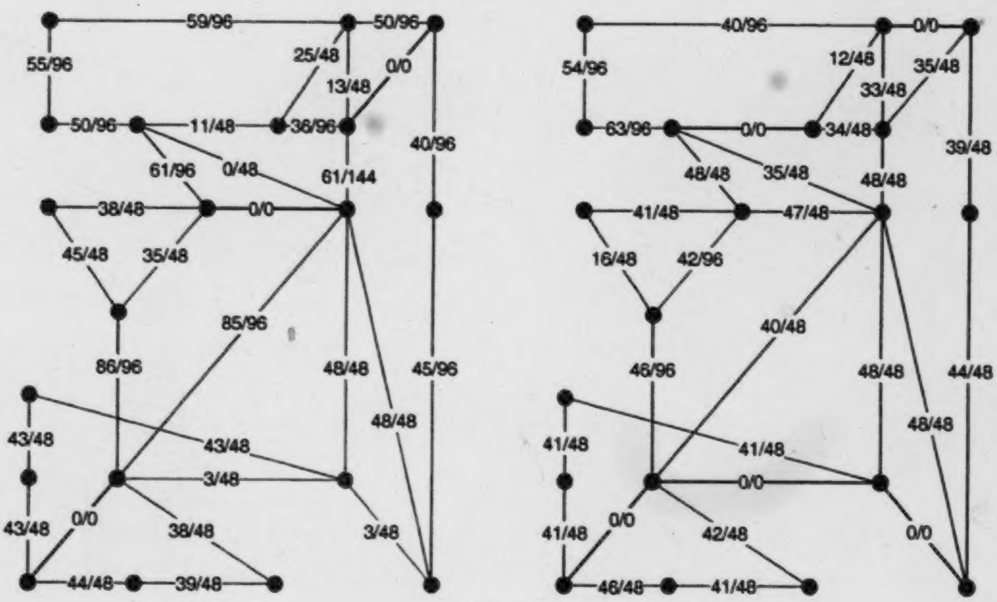


Figure 11.7. Tabu Search solution for Net20.

A breakdown of the working load and capacity on each span of these designs is shown in Fig.

11.8. For the initial design, the sum of the working load on all spans is 1,147 DS3-hops and the total capacity placed is 1,920 DS3-hops. This gives an average ring utilization (or fill) of 60%. Because the total demand for Net20 is 348 DS3s, the average number of hops per demand is $1147/348 = 3.3$. For the Tabu Search solution, the sum of the working load on all spans is 1,065 DS3-hops and the total capacity placed is 1,440 DS3-hops. Although the average number of hops per demand is only slightly less than in the initial design, the total capacity is a full 25% lower. This results in an average ring fill of roughly 75%.

Figure 11.8 also shows that the initial solution results in three spontaneous span eliminations, while the TS solution has five span eliminations. That is, five spans in the TS solution carry no demands and are not covered by a ring.


59/96
50/96
55/96
25/48
0/0
13/48
50/96
11/48
36/96
40/96
61/96
0/48
61/144
38/48
0/0
45/48
35/48
85/96
86/96
48/48
45/96
43/48
43/48
48/48
3/48
43/48
0/0
38/48
3/48
44/48
39/48
40/96
0/0
54/96
12/48
35/48
33/48
63/96
0/0
34/48
39/48
48/48
35/48
48/48
41/48
47/48
16/48
42/96
40/48
46/96
48/48
44/48
41/48
41/48
48/48
0/0
41/48
0/0
42/48
0/0
46/48
41/48

(a) (b)

Figure 11.8. Span utilization (working load/capacity) in DS3s for Net20 (a) initial solution and (b) Tabu Search solution.

Based on the working load for the initial design, the lower bound on the required capacity is 1,680 DS3-hops, while the actual capacity is 1,920 DS3-hops. In fact at least 5 ring modules in the initial design are completely empty. In contrast, the lower bound on the required capacity for the TS design is 1,344 DS3-hops, seven ring modules fewer than the initial design. Furthermore, the actual capacity in the TS design is only 96 DS3-hops (or 2 ring modules) greater than the lower bound.

A histogram of the slack capacity on each span in both designs is also shown in Fig. 11.9. The slack (or unused) capacity is equal to the aggregate capacity on a span less its working load. The histogram in Fig. 11.9 shows that roughly 30% percent of the spans in the initial design have a slack capacity that equals or exceeds one ring module (i.e., 48 DS3s). This represents a significant waste of transmission capacity. In contrast, roughly 80% percent of the spans in the TS solution have a slack capacity of less than 20 DS3s. A detailed examination of the results shows that the initial solution has a total slack of 773 DS3s, while the the TS solution has a total slack of 375 DS3s (less than half the initial solution).

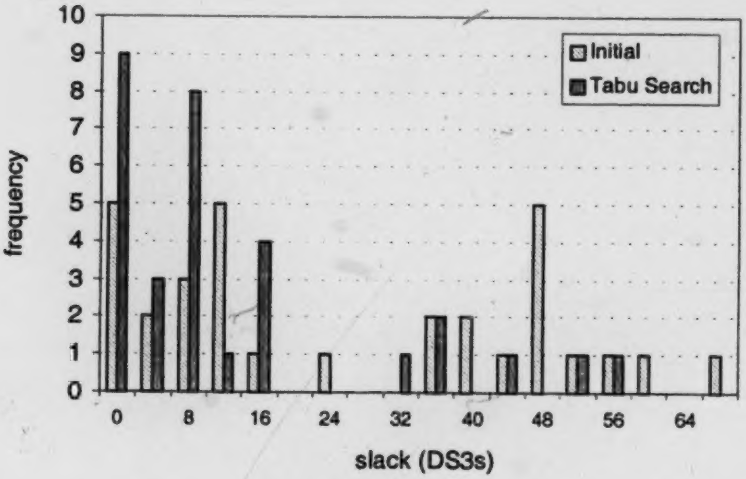


Figure 11.9. Histogram of the slack capacity for Net20.

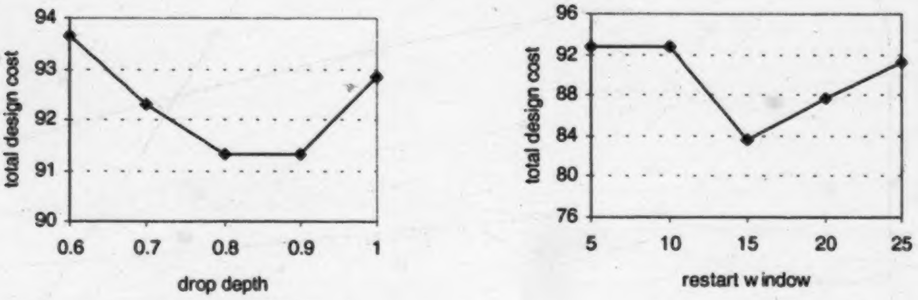
The results for different Add and Drop Tenure parameter settings are listed in Table 11.2. These results show that, with the exception of one test case, the total design cost is completely insensitive to changes in the Add Tenure. An examination of the detailed log for each TS run reveals that the solution quickly converges to a design containing only three rings. In this situation, an Add Tenure of three or more has the same effect on the search trajectory because the drop moves for all of the rings in the solution will be tabu, having just been added to the solution. Although there are minor variations in some of the search trajectories, the same best solution is found for almost all values of Add Tenure.

In contrast, changes in the Drop Tenure does have an effect on the total design cost. The lowest cost designs are consistently achieved with a Drop Tenure of only two iterations. This suggests that the simple search procedure developed here is relatively immune to cycling. One explanation for this observation is as follows. When a ring is dropped from the solution and another ring(s) is added in its place, the transport efficiency of the dropped ring will remain relatively low until the ring(s) that displaced it are removed from the solution. This is because most, if not all, of the demand served by the dropped ring is already served by the new ring(s). Thus, there is a natural tendency for rings that are dropped from being added back to the solution. In this situation, a shorter Drop Tenure can be helpful because it allows the search to converge more quickly on a local optimum.

Table 11.2: Total Design Cost for Several Add and Drop Tenures

	Add Tenure					
Drop Tenure	2	3	4	6	8	10
2	83.43	83.43	83.43	83.43	83.43	83.43
3	93.28	93.28	93.28	93.28	93.28	93.28
4	92.86	92.86	92.86	92.86	92.86	92.86
6	93.29	93.28	93.28	93.28	93.28	93.28
8	91.37	91.37	91.37	91.37	91.37	91.37
10	84.93	91.01	91.01	91.01	91.01	91.01

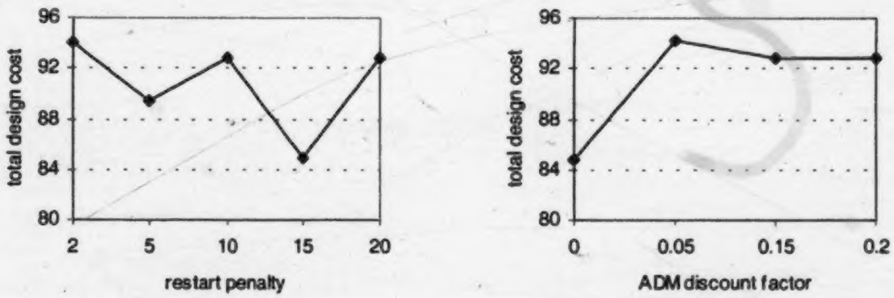
The results also show that changes in the penalty factor for add and drop moves had virtually no effect on the search trajectory for the range of values tested. The effect of the drop depth parameter on the search is shown in Fig. 11.10(a). This figure shows that the lowest cost designs are obtained with a drop depth in the range from 0.8 to 0.9. One explanation for this behaviour is that a lower drop depth makes more demand available for packing into the next candidate ring that is added during the Constructive Phase. This improves the likelihood that the next candidate ring will be filled to capacity. If the drop depth is too low, however, more than one candidate ring may be required and the combined efficiency of both rings may suffer. With even lower drop depths, the search may not converge to a local minima because the solution may change too dramatically from one iteration to the next. In other words, there would be no opportunity for fine tuning the design. An interesting strategy would be to vary the drop depth during the search to alternate between periods of intensification and diversification. Figure 10.11(b) shows the best solution cost over a range of different restart window sizes. These results show that a restart window of 15 iterations provides the best solutions for the current problem.



(a) (b)

Figure 11.10. Plot of total design cost vs. (a) drop depth and (b) restart window.

The effect of the restart penalty on the best solution cost is also shown in Fig. 11.11(a). Based on this data, there does not appear to be a strong relationship between the restart penalty and the best solution cost. The best solution cost is also shown as a function of the ADM discount factor (for demand packing) in Fig. 11.11(b).



(a) (b)

Figure 11.11. Plot of total design cost vs. (a) restart penalty and (b) ADM discount factor.

This figure indicates that the best solution is found when the ADM discount factor is set to zero. This finding is contrary to the demand packing results presented in Chapter 8. One possible explanation for this difference is that higher ADM discount factors generally reduce capacity efficiency, making it more difficult to serve all demands during the destructive phase. As a result, fewer rings may be dropped from the solution than would have otherwise been possible.

Although the above results give some indication of the best parameter settings for the present problem, finding the empirically best parameter settings over a wide range of problem instances is one of the aspects that deserves more study (as is normally the case with any meta-heuristic).

11.8 Summary

In this chapter, we developed a simple TS procedure for the multi-ring network design problem. Despite the relative simplicity of this procedure, the it performs extremely well in the test cases considered here. In the next chapter, we compare the performance of all tests methods.

Table 11.3: Detailed Computational Results for Tabu Search Meta-Heuristic

Test Case	Net-work	Tech.	Restart Events		Iterations	Runtime (sec.)	Initial Solution	Best Solution
			Gap Exceeded	Previous Solution				
1	Net15	4B12	8	0	100	2,471	71.28	57.02
2	"	4B48	7	2	100	2,231	58.89	54.41
3	"	4B12, 4B48	7	1	93	7,316	55.48	49.31
4	Net20	4B12	9	0	100	2,284	117.20	108.77
5	"	4B48	9	0	100	2,451	86.43	81.67
6	"	4B12, 4B48	6	0	71	7,543	84.43	81.71
7	Net32	4B48	5	6	100	312	786.88	745.42
8	"	4B192	6	4	100	369	655.88	593.92
9	"	4B48, 4B192	8	0	100	595	617.90	525.14
10	Net43	4B48	1	0	39	15,340	1,323.6	1,240.0
11	"	4B192	0	0	10	7,922	907.15	890.71
12	"	4B48, 4B192	0	0	9	9,576	877.71	835.99

Table 11.4: Design Statistics for Tabu Search Meta-Heuristic

Test Case	Net-work	Number of Rings			Number of ADMs			Regen.	Fibre (km)	Avg. Fill (%)	Inter-ring Transit.	Avg. Hops/ Route	Spans Elim.	Total Working (DS3-hops)	Total Capacity (DS3-hops)	Total Cost	Runtime (sec.)
		4B12	4B48	4B192	4B12	4B48	4B192										
1	Net15	5	-	-	41	-	-	4	803.3	81.3	13	2.18	5	449	552	57.02	2,471
2	"	-	2	-	-	21	-	0	401.2	59.5	1	3.74	9	771	1,296	54.41	2,231
3	"	2	1	-	14	10	-	1	342.3	82.0	31	2.63	6	541	660	49.31	7,316
4	Net20	10	-	-	71	-	-	22	2,374.4	79.9	41	2.56	0	892	1,116	108.77	2,284
5	"		3	-	-	28	-	2	794.0	74.0	39	3.06	5	1,065	1,440	81.67	2,451
6	"	0	3	-	0	27	-	9	1,022.0	69.1	34	3.43	5	1,194	1,728	81.71	7,543
7	Net32	-	7	-	-	53	-	321	108,204	58.1	165	5.99	3	2,120	3,648	745.42	312
8	"	-	-	6	-	-	35	212	74,724	19.8	202	6.03	5	2,133	10,752	593.92	369
9	"	-	2	3	-	14	21	202	67,788	36.2	161	6.49	7	2,296	6,336	525.14	595
10	Net43	-	27	-	-	201	-	207	91,176	82.2	2,148	3.75	1	9,471	11,520	1,240.0	15,340
11	"	-	-	11	-	-	89	71	35,912	56.4	2,147	4.21	19	10,620	18,816	890.71	7,922
12	"	-	2	9	-	7	71	74	37,668	64.1	2,086	4.40	17	11,111	17,328	835.99	9,576

12 Comparative Discussion and Interpretation

12.1 Introduction

This chapter compares the performance of the design methods presented in the preceding chapters. We begin by comparing the relative performance of the design methods in terms of solution quality and runtime. The correlation between several design attributes and total design cost is also calculated to draw insights into the most dominant factors affecting solution quality. In Section 12.3 we consider the absolute performance of the methods by comparing the empirical results with lower bounds established using the procedure described in Chapter 7. We also derive point estimates and confidence intervals for the optimal solution values using the statistical techniques described in Chapter 7.

12.2 Solution Quality

The relative performance of the design methods is summarized in Figure 12.1. For each design method, this figure shows the minimum, maximum and mean "gap" between the current and best solutions over all test cases. The "gap" is the percentage difference in the total design cost between the current solution (i.e., the solution obtained by the indicated design method) and the best solution found by any of the methods. The total design cost for each test case and design method are also listed in Table 12.2 at the end of this chapter.

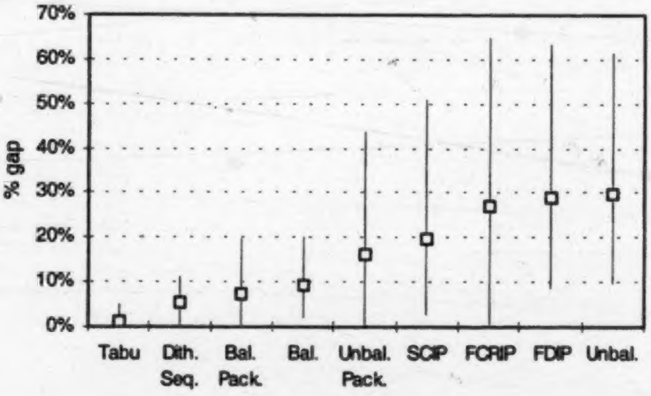


Figure 12.1. Relative performance of design methods (% gap) relative to the best solution for all test cases.

These results show that the Tabu Search (TS) meta-heuristic consistently produces network designs that are the best solutions, or are within a few percent of the best found solutions. The detailed results in Table 12.2 also show that the TS meta-heuristic generates the best solution in 8 out of the 12 test cases. In the remaining test cases, the total design costs of the TS solutions are within 3% of the best solution. The next most successful method, in terms of total design cost, is Dithered Sequencing (DS). On average, DS generates solutions that are within about 5% of the min-cost solution. However, the runtime for DS is significantly longer than some of the other methods.

Although the FCRIP formulation produces the best solution in two of the smaller test cases, performance deteriorates quickly as the problem size increases (due entirely in practice to run-time limits that curtail its ability to approach the optimum). This explains the large range in the gap values shown in Fig. 12.1. The poorest designs were produced by the baseline RingBuilder algorithm with greedy (unbalanced) ring loading, which produces designs that are about 30% more costly, on average, than the best solutions. In a few cases, the cost of the designs produced by the baseline RingBuilder algorithm are up to 60% higher than the best solutions. The performance of the RingBuilder algorithm improves dramatically, however, when the demand packing improvement heuristic is used.

12.3 Runtimes

The total runtimes for each design method and test case are listed in Table 12.3 at the end of this chapter. Note that the runtime results for the mathematical programming approaches are not directly comparable with those from the heuristic methods because they solve a simplified version of the problem. Nonetheless, the results show that the runtimes for the heuristic methods scale best with larger problem sizes. Here, the "problem size" is taken as the product of the number of demand pairs and the number of candidate rings under consideration. This is based on an intuitive appreciation that the product of the number of demand pairs and candidate rings underlies the basic complexity of the problem. Figure 12.2 shows a log-log plot of the runtimes for the baseline RingBuilder algorithm (i.e., unbalanced ring loading without demand packing or dithered sequencing) and the Tabu Search meta-heuristic. A (linear) regression line is also fitted to the data from each method and plotted in Fig. 12.2. From this plot, the relationship between the problem size, n, and the runtimes, $f(n)$, can be estimated by the following expression

$$f(n) = k \cdot n^{m} \qquad (1)$$

where m is equal to the slope of the regression line and k is a constant. Note that a slope $m = 1$ indicates a linear relationship between runtimes and problem size. The slope of the regression line for the baseline RingBuilder algorithm is 0.77, which suggests that its runtime is roughly a linear function of the problem size. Similar results are observed for the RingBuilder algorithm with balanced ring loading and the Dithered Sequencing approach. The regression line for the Tabu Search meta-heuristic also indicates that its runtime increases linearly with problem size. The runtimes for the TS meta-heuristic are normalized to 100 iterations because the number of iterations performed within the two hour time limit on execution is not the same for all test cases. Note that these runtimes also include the time required to generate the initial starting solution.

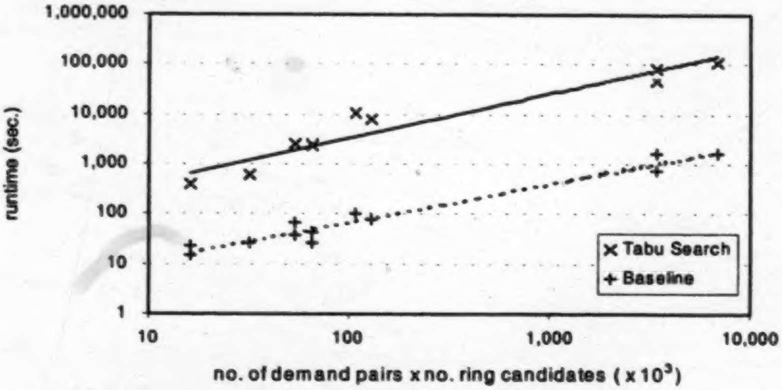


Figure 12.2. Plot of runtime vs. problem size for baseline RingBuilder heuristic and Tabu Search meta-heuristic.

In contrast, the runtimes for the SCIP formulation increases exponentially with problem size, as shown in Figure 12.3. This is expected because the branch-and-bound technique used by the CPLEX MIP solver can continue long after the best solution has been found [CPL98].

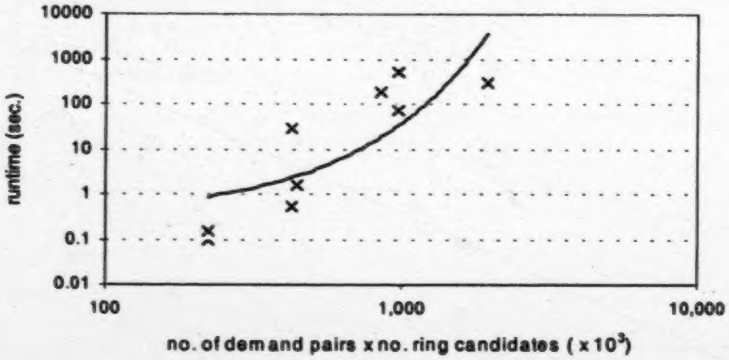


Figure 12.3. Plot of runtime vs. problem size for SCIP formulation.

In the worst case, the CPLEX MIP Solver may perform an exhaustive search of the branch-and-bound tree. Even if all the decision variables are binary, the branch-and-bound tree may be as large as 2^n, where n is the number of variables.

Similar plots for the FCRIP formulation are not shown because the runtime limit was exceeded in all test cases before a provably optimal solution could be found. Likewise, plots for the FDIP are not shown because the size (i.e., number of variables) of the problems is almost the same in all test cases. Although the FDIP formulation generally offers the lowest runtimes, these figures do not include the time required to populate the candidate ring set, which can be significant.

12.4 Significance of Design Attributes

For insights into the relative importance of various design attributes on solution quality, we calculated the correlation coefficient between these attributes and the total design cost. These results are listed in Table 12.4. The star plot in Figure 12.2 summarizes the results for two test cases: test case 5 (net20, 4B48) and test case 7 (net32, 4B48).

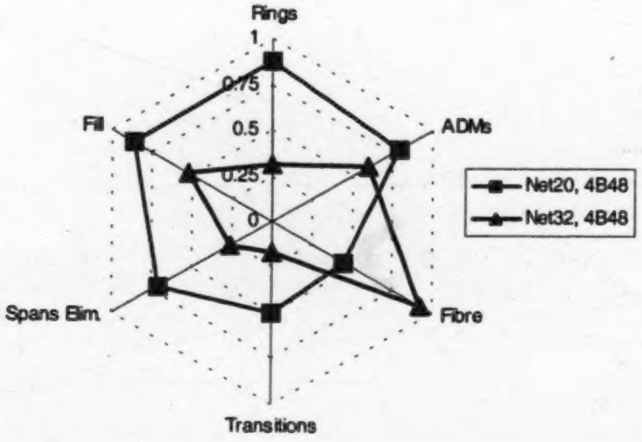


Figure 12.4. Star plot of the correlation squared between several design attributes and total design cost for test cases 5 and 7.

This figure shows the correlation squared between the total design cost and the number of rings, number of ADMs, total fibre mileage, number of transitions, number of eliminated spans and the average network fill. For test case 5 (Net20, 4B48), this plot shows that the total design cost is closely correlated with the number of rings, the average network fill and the number of ADMs. The total fibre mileage, on the other hands, plays a less influential role in determining the total design cost. This is to be expected because the distances in Net20 are relatively short and, therefore, the distance dependent costs account for a relatively small fraction (< 10%) of the total design cost. In contrast, the total fibre mileage is strongly correlated with the total design cost in test case 7 (Net32, 4B48). This is because the span distances in Net32 are much longer and, therefore, the distance-dependent costs make up the majority (> 75%) of the total design cost. These results are representative of the other test cases. One implication of these results is that for long-haul networks, design methods that focus primarily on minimizing the total fibre mileage are likely to produce superior results. Whereas, in metropolitan area networks, the design methods must take a more balanced approach. Of course, these generalizations are sensitive to changes in the cost model.

12.5 Lower Bounds

Lower bounds on the number of ADMs, ring modules, regenerators, fibre mileage and total design cost are listed in Tables 12.5. These results are based on the lower bounding procedure using shortest-path routing. Also listed in Table 12.5 is the total design cost for the best solution cost and the gap (in percent) between the lower bound and the best solution. These results show that in most cases there is a large gap (about 35% on average) between the lower bound and the best solution but the results vary greatly. In test cases 11 (Net43, 4B192) and 12 (Net43, 4B48 + 4B192), for example, the cost of the best solution is roughly 75% higher than the lower bound. Whereas, in test case 7 (Net32, 4B48) the cost of the best solution is within 5% of the lower bound based on shortest-path routing.

A detailed comparison of the results reveals that the lower bounding procedure consistently under estimates the number of ADMs required in the design. One reason for this is that the procedure does not account for the ADMs and ADM add/drop interfaces required for inter-ring transitions. However, the procedure does provide a fairly tight lower bound on the total fibre mileage for those design methods that do not reroute demands during the design synthesis process (i.e., the basic algorithm without demand packing and the SCIP formulation). For example, in five out of twelve designs generated using the SCIP formulation, the actual fibre mileage is the same as the lower bound. In the remaining seven test cases, the actual fibre mileage is roughly 8% higher than the lower bound.

An examination of the detailed results for the other design methods that allow demands to be rerouted during the synthesis process shows that in some cases the total fibre mileage is actually lower than the lower bound based on shortest path routing. For example, in test case 2 (Net15, 4B48), the total fibre mileage is 23% lower than that from the lower bounding procedure.

These results prompted the reformulation of the lower bounding procedure to simultaneously optimize the routing of demands, as described in Chapter 7. To obtain lower bounds using this formulation, we generate three shortest paths for each demand pair and solve the resulting problem instance using the CPLEX MIP Solver (CPL98). The lower bounds from this formulation are given in Table 12.6. Note that results are not presented for Net43 because the set of shortest paths could not be generated. This is because the algorithm used here for finding the shortest paths is not particularly efficient and could not find the set of paths within the available computer memory. These results show that the total number of ring modules and fibre mileage can be significantly less with routing optimization, as expected. For example, in test case 2 (Net15, 4B48) the lower bound on the total fibre mileage is 34% lower than the bound obtained with shortest path routing. The difference in the bound on total design cost, however, is less remarkable. On average, the bounds on total design cost obtained with route optimization are roughly 7% lower than those obtained with shortest path routing. In any case, these bounds provide a rough indication of the quality of the solutions generated by the design methods. For example, Table 12.6 shows that the best solution for test case 8 (Net32, 4B192) lies within at least 15.6% of the optimal solution. The results in the next section indicate that the best solutions are actually much closer to the optimal solutions than the lower bounds suggest.

12.6 Statistical Inference

Histograms of the total design cost for all solutions are shown in Figures 12.5 through 12.6. These histograms include results for all design methods including intermediate solutions obtained from the Tabu Search and Dithered Sequencing methods. In most cases, the sample size is greater than 100. For test network Net43, the sample size was smaller because of the long runtimes required for each solution. This explains the relative coarseness of the histograms for these test cases. A Weibull probability density function is also plotted in these figures based on estimates of the Weibull shape, scale and location parameters. The point and 95% confidence interval estimates of the global optimum solution are presented in Table 12.1. These results are obtained using the statistical inference techniques outlined in Chapter 7. Table 12.1 also lists the correlation squared (R^2) and the gap (in percent) between the upper and lower values of the interval estimate. The correlation squared measures of the goodness-of-fit between the empirical results and the fitted Weibull distribution from which the estimates are obtained. These results suggest that the best solutions from the design methods developed here are within about 12% of the globally optimal solution. These estimates, however, are based on the assumption that the solutions obtained by the various methods are independent. Although this assumption requires further testing, the estimates provided here offer at least some indication of the absolute solution quality.

Table 12.1: Point and Interval Estimates of Global Optimal Solutions

Network	Tech.	Point Estimate	R^2	Interval Estimate	Gap (%)
Net15	4B12	51.2	.961	[48.9, 54.5]	11.5
"	4B48	52.9	.951	[52.0, 54.4]	4.6
"	4B12, 4B48	49.0	.972	[48.7, 49.0]	0.6
Net20	4B12	104.3	.959	[101.7, 105.8]	4.0
"	4B48	78.2	.989	[76.4, 81.7]	6.9
"	4B12, 4B48	81.4	.976	[80.3, 81.7]	1.7
Net32	4B48	657.6	.961	[626.5, 680.6]	8.6
"	4B192	561.4	.988	[548.5, 568.7]	3.7
"	4B48, 4B192	511.0	.949	[491.1, 525.1]	6.9
Net43	4B48	1,228	.937	[1218, 1240]	1.8
"	4B192	831.5	.908	[807.3, 852.4]	5.6
"	4B48, 4B192	829.6	.964	[816.7, 836.0]	2.4

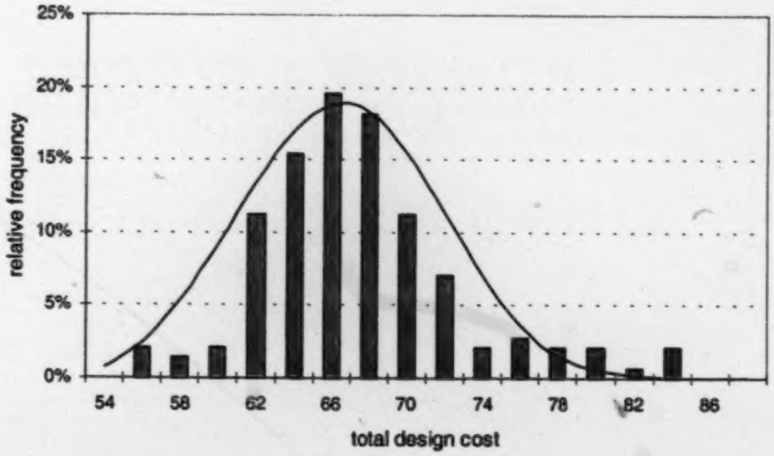


Figure 12.5. Histogram of total design cost for test case 1 (Net15, 4B12).

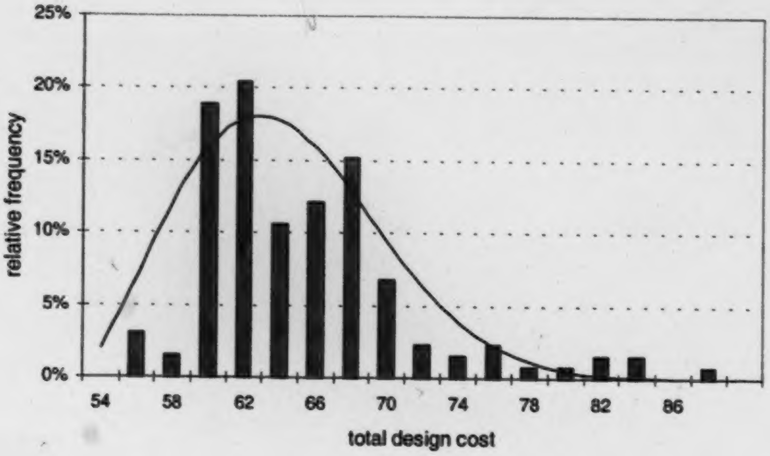


Figure 12.6. Histogram of total design cost for test case 2 (Net15, 4B48).

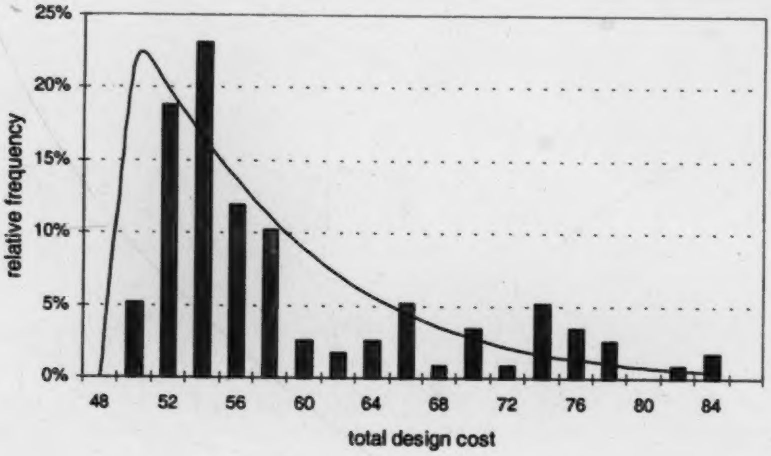


Figure 12.7. Histogram of total design cost for test case 3 (Net15, 4B12 + 4B48).

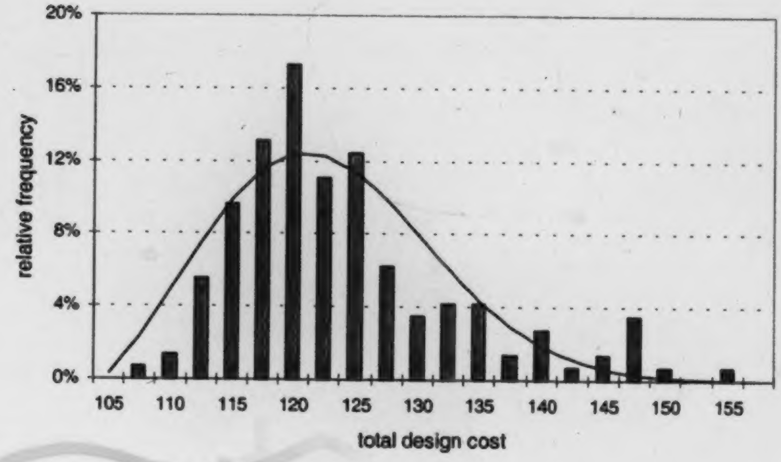


Figure 12.8. Histogram of total design cost for test case 4 (Net20, 4B12).

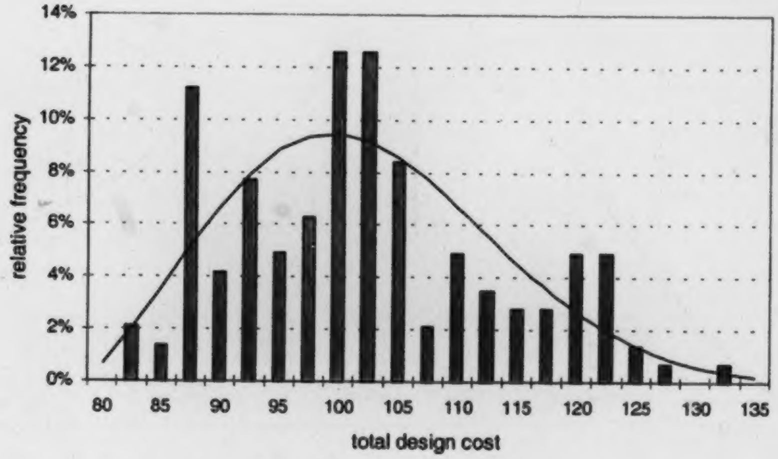


Figure 12.9. Histogram of total design cost for test case 5 (Net20, 4B48).

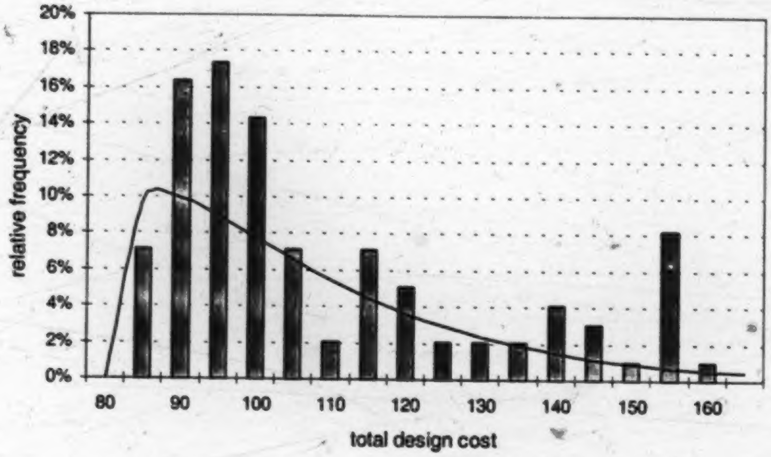


Figure 12.10. Histogram of total design cost for test case 6 (Net20, 4B12 + 4B48).

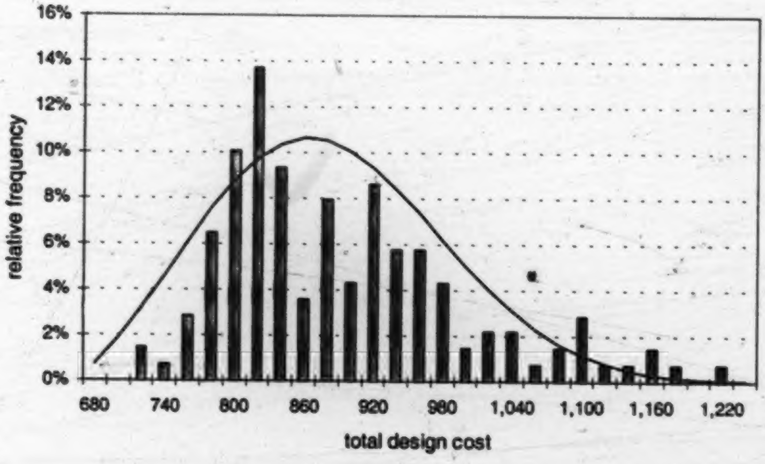


Figure 12.11. Histogram of total design cost for test case 7 (Net32, 4B48).

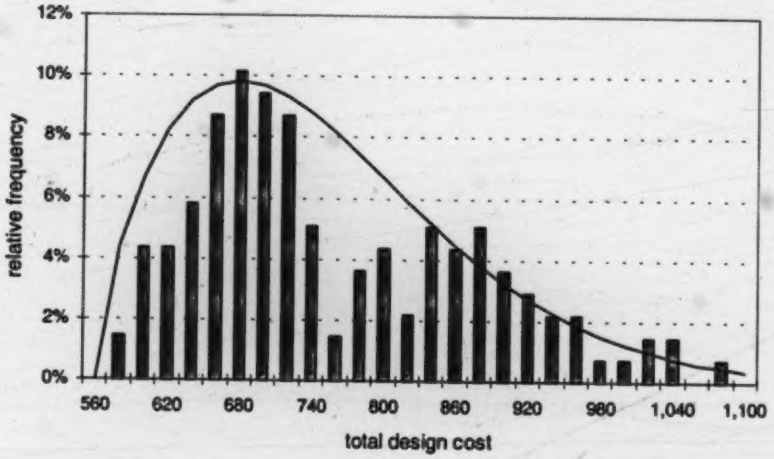


Figure 12.12. Histogram of total design cost for test case 8 (Net32, 4B192).

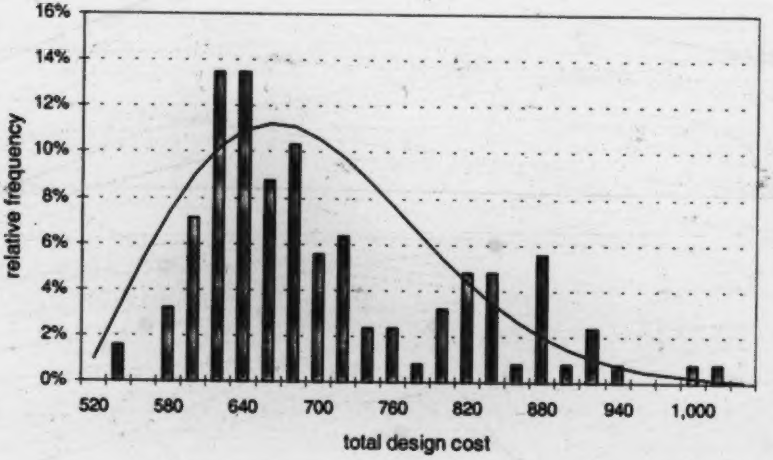


Figure 12.13. Histogram of total design cost for test case 9 (Net32, 4B48 + 4B192).

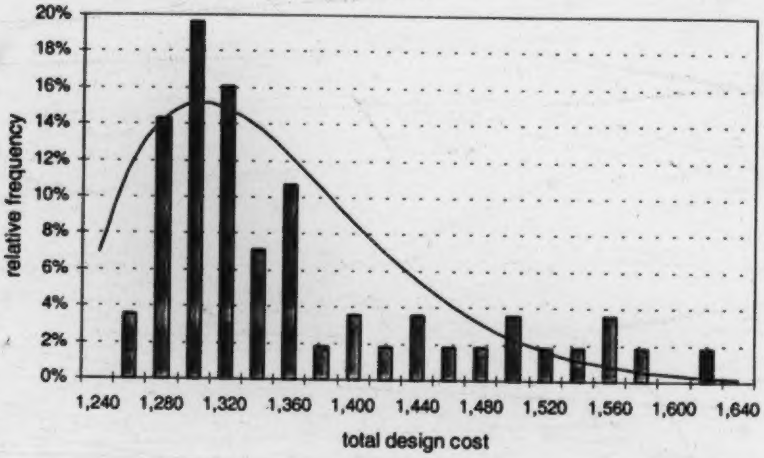


Figure 12.14. Histogram of total design cost for test case 10 (Net43, 4B48).

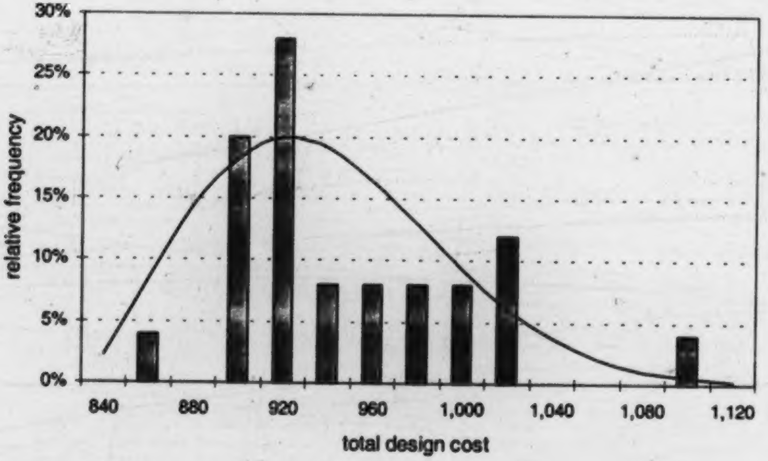


Figure 12.15. Histogram of total design cost for test case 11 (Net43, 4B192).

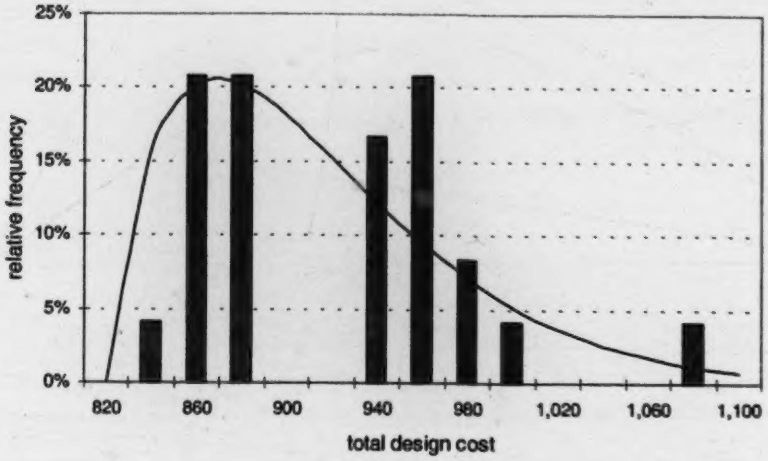


Figure 12.16. Histogram of total design cost for test case 12 (Net43, 4B48 + 4B192).

12.7 Summary

In this chapter, we compared the performance of the design methods in terms of solution quality and runtime. The results show that, overall, the Tabu Search meta-heuristic provides the best solutions in the majority of test cases. Furthermore, the results also indicate that the runtime of the Tabu Search meta-heuristic increases linearly with the product of the number of demand pairs and candidate rings. Using statistical inference it has also been shown that the best solutions found using the design methods developed here lie within about 12% of the estimated optimal solution. Although further work is required to validate the independence assumptions, these estimates provide at least some indication of the absolute solution quality of the methods developed here. To the best of our knowledge, this is the first application of statistical inference techniques to the multiring network design problem.

Table 12.2: Summary of Total Design Cost Results

Network	Tech.	Unbal.	Bal.	Basic Packing	Bal. Packing	Dith. Seq.	SCIP	FCRIP	FDIP	Tabu
Net15	4B12	68.71	66.58	63.45	62.41	61.48	67.45	55.35*	70.00	57.02
"	4B48	87.86	59.87	78.23	55.44	58.41	82.11	77.12*	63.62	54.41
"	4B12, 4B48	68.71	52.26	63.45	52.75	50.11	65.73	81.27*	55.77	49.31
Net20	4B12	118.77	117.20	113.00	118.90	113.91	120.36	105.79*	126.85	108.77
"	4B48	119.31	86.43	98.04	82.41	86.96	104.46	92.56*	111.10	81.67
"	4B12, 4B48	117.74	84.43	106.11	82.41	83.68	100.06	110.19*	99.46	81.71
Net32	4B48	910.12	786.88	803.52	787.32	709.64	876.72	912.46*	1,044.2	745.42
"	4B192	660.68	655.88	568.72	667.46	595.56	608.42	693.80*	928.94	593.92
"	4B48, 4B192	645.38	617.90	532.72	629.48	583.42	581.72	727.32*	847.60	525.14
Net43	4B48	1,358.1	1,288.7	1,283.6	1,244.4	1,274.8	1,272.8*	-	1,488.8*	1,240.3
"	4B192	1,097.4	907.15	1,019.1	909.33	891.47	961.29*	-	967.53	890.71
"	4B48, 4B192	1,074.0	877.15	985.97	859.47	920.47	941.33*	-	945.93	835.99

*runtime limit exceeded before an optimal solution was found. Runtimes include total CPU time for all four processors.

Table 12.3: Summary of Runtime Results (sec.)

Network	Tech.	Unbal.	Bal.	Basic Packing	Bal. Packing	Dith. Seq.	SCIP	FCRIP	FDIP	Tabu
Net15	4B12	45	106	40	87	954	71.3	5,452*	0.5	2,471
"	4B48	26	50	26	46	450	522.3	6,433*	0.02	2,231
"	4B12, 4B48	78	104	72	100	936	308.9	6,471*	0.19	7,316
Net20	4B12	69	130	63	124	1,170	0.53	7,864*	70.4	2,284
"	4B48	37	51	28	49	459	28.63	8,333*	0.10	2,451
"	4B12, 4B48	105	99	78	90	891	186.7	6,402*	0.16	7,543
Net32	4B48	23	24	26	27	216	0.10	8,284*	0.07	312
"	4B192	15	19	25	18	171	0.15	8,642*	0.01	369
"	4B48, 4B192	26	31	31	32	279	1.63	8,265*	0.02	595
Net43	4B48	1,610	6,552	1,655	6,520	58,968	4,508*	-	14,342*	15,340
"	4B192	758	2,932	659	2,980	26,388	6,464*	-	7.15	7,922
"	4B48, 4B192	1,607	5,484	1,408	5,478	49,356	5,424*	-	9.52	9,576

*runtime limit exceeded before an optimal solution was found. Runtimes include total CPU time for all four processors.

Table 12.4: Correlation between Design Attributes and Total Design Cost

Test Case	Net-work	Samples	Number of Rings			Number of ADMs			Regen.	Fibre (km)	Avg. Fill (%)	Inter-ring Transit.	Spans Elim.	Total Working (DS3-hops)	Total Capacity (DS3-hops)
			4B12	4B48	4B192	4B12	4B48	4B192							
1	Net15	44	-0.2416	-	-	0.8460	-	-	0.0594	0.7569	-0.2896	0.1818	-0.3441	0.5428	0.8889
2	"	31	-	0.8345	-	-	0.9222	-	0.1066	0.7528	-0.8047	0.5572	-0.9236	-0.4967	0.7645
3	"	26	.3869	0.3057	-	0.4819	0.0586	-	0.0740	0.7858	-0.6288	0.2781	-0.4912	-0.1679	0.5873
4	Net20	42	-0.0349	-	-	0.8494	-	-	0.3184	0.6028	-0.7804	-0.1829	0.0077	0.2710	0.6200
5	"	39	-	0.9392	-	-	0.8917	-	0.5250	0.6673	-0.9288	0.7091	-0.8416	-0.7615	0.7445
6	"	24	0.6729	0.4791	-	0.6762	0.3787	-	0.1998	0.3116	-0.5966	0.8071	-0.4313	-0.7608	-0.4526
7	Net32	30	-	0.5552	-	-	0.7769	-	0.9288	0.9604	-0.7195	0.4097	-0.5127	-0.1115	0.8888
8	"	26	-	-	0.0434	-	-	0.7323	0.7207	0.8348	-0.8089	-0.1247	-0.2687	-0.5040	0.7260
9	"	21	-	0.0976	0.0992	-	0.0883	0.1294	0.8761	0.9060	-0.3277	-0.0912	-0.6815	-0.0182	0.4255
10	Net43	18	-	0.4770	-	-	0.7069	-	0.4313	0.8008	-0.8766	-0.2660	0.0109	-0.2399	0.9311
11	"	16	-	-	0.8336	-	-	0.8066	0.4967	0.6148	-0.7708	0.6572	-0.5536	-0.3513	0.8391
12	"	16	-	0.8093	0.0596	-	0.7761	0.2706	0.5631	0.7658	-0.4859	0.6535	-0.6081	-0.6066	-0.0330

Table 12.5: Lower Bounds based on Shortest Path Routing

Test Case	Net-work	Number of ADMs			Number of Ring-Spans			Regen.	Fibre (km)	Total Cost	Best Solution	Gap (%)
		4B12	4B48	4B192	4B12	4B48	4B192					
1	Net15	24	-	-	50	-	-	26	770.5	43.35	55.35	27.7
2	"	-	15	-	-	34	-	19	521.3	46.71	54.41	16.5
3	"	9	6	-	12	9	-	19	521.3	37.71	49.31	30.8
4	Net20	38	-	-	89	-	-	51	2,234	76.77	105.79	37.8
5	"	-	20	-	-	40	-	20	992.8	66.36	81.67	23.1
6	"	7	13	-	3	37	-	20	992.8	59.36	81.71	37.6
7	Net32	-	29	-	-	83	-	343	106,968	679.14	709.64	4.5
8	"	-	-	26	-	-	47	209	67,564	501.32	568.72	13.4
9	"	-			-	8	39	209	67,564	455.32	525.14	15.3
10	Net43	-	86	-	-	241	-	290	80,436	758.43	1,240.3	63.5
11	"	-	-	43	-	-	109	128	36,792	507.81	890.71	75.4
12	"	-	0	43	-	0	109	128	36,792	507.81	835.99	75.4

Table 12.6: Lower Bounds with Route Optimization

Test Case	Net-work	Number of ADMs			Number of Ring-Spans			Regen.	Fibre (km)	Total Cost	Best Solution	Gap (%)
		4B12	4B48	4B192	4B12	4B48	4B192					
1	Net15	24	-	-	40	-	-	16	580.4	40.40	55.35	37.0
2	"	-	15	-	-	25	-	10	344.3	44.02	54.41	23.6
3	"	9	6	-	3	21	-	9	344.3	34.82	49.31	41.6
4	Net20	38	-	-	71	-	-	33	1,696	70.48	105.79	50.1
5	"	-	20	-	-	25	-	5	674.8	61.77	81.67	32.2
6	"	7	13	-	0	27	-	7	674.8	55.17	81.71	48.1
7	Net32	-	29	-	-	63	-	323	89,824	589.42	709.64	20.4
8	"	-	-	26	-	-	51	213	65,532	491.96	568.72	15.6
9	"	-			-	26	25	205	65,532	444.36	525.14	18.2
10	Net43	-	86	-	-	?	-	?	?	?	1,240.3	?
11	"	-	-	43	-	-	?	?	?	?	890.71	?
12	"	-	0	43	-	?	?	?	?	?	835.99	?

13 Concluding Discussion

13.1 Introduction

This chapter concludes the thesis with a review of the main developments, a summary of the research contributions and recommendations for further research.

13.2 Review of Thesis

The primary goal of this work was to develop improved methods for solving the multi-ring network design problem by better understanding the factors that influence the design problem and by applying advanced heuristic and mathematical programming techniques. In Chapter 1 we saw that network survivability is an increasingly important consideration in the design and operation of transport networks. Growing demand, higher-capacity systems and an increased reliance on telecommunications services are all contributing factors in the need for improved network survivability. Survivable rings provide a robust and cost-effective means of protecting these networks against cable cuts and other faults, but the design of ring-based transport networks is a difficult optimization problem. To date, most methods proposed for this problem are either over-simplified or offer no guarantees on optimality. Yet there is strong economic motivation for even modest improvements in the performance of the current design methods given the large capital costs associated with these networks. We have shown that significant improvements towards optimality can be achieved with new optimization techniques.

Chapters 2 provided a tutorial overview of some basic concepts and terminology from several related fields of study including graph theory, mathematical programming and complexity theory. In Chapter 3 we introduced a generic view of transport networks and discussed the two most common transport technologies in use today: PDH and SONET. We also defined network survivability in precise terms and described survivable rings in detail. These preliminary chapters provided background information used throughout the remainder of the thesis.

In Chapter 4 we defined and developed IP formulations for several variants of the ring loading problem — an important subproblem of the multi-ring network design problem. We also studied the effect of different loading policies and technology choices on the loading efficiency of bidirectional line-switched rings. The results of this study showed that channel interchange gives almost no advantage in terms of loading efficiency over an optimally planned channel assignment solution. This is a particularly relevant finding for the multi-ring network design problem because it means that full channel interchange can be assumed with little or no impact on the overall solution quality. This greatly simplifies the multi-ring network design problem because actual channel assignment decisions, if required, can always be made off-line after the design is complete. This study also shows that significant gains in loading efficiency can also be obtained by allowing demand bundles to be split between the two directions around the ring, even though this is generally not a common practice.

Chapter 5 provided a formal definition the multi-ring network design problem and assessed its computational complexity. Several other design considerations were discussed. This material served as framework for reviewing the prior work on the ring network design problems presented in Chapter 6.

Chapter 7 presented the characteristics of real transport networks used later in specific studies. The modelling assumptions used to evaluate the performance of the design methods developed in subsequent chapters were also described in detail. Metrics were developed to quantify the results and the methods for assessing the relative and absolute performance of each design method. These methods included two new lower bounding procedures for the multi-ring network design problem and statistical inference techniques for obtaining both point and interval estimates of the optimal solution.

In Chapters 8 we describe the sophisticated network planning tool, called RingBuilder Interactive, that was developed to support this work. In addition to a basic greedy heuristic algorithm, several improvement heuristics were developed including a balanced ring loading algorithm, a demand packing algorithm and a dithered sequencing meta-heuristic.

Chapter 9 describes the comparative study method and provides results for the basic RingBuilder algorithm and improvement heuristics developed in Chapter 8. Overall, the experimental results showed that significant savings in total design cost can be achieved using several of the improvement heuristics. Savings of up to 32% relative to the benchmark solution were obtained using the balanced ring loading and demand packing algorithms.

In Chapter 10, we developed three new mathematical programming formulations for the multi-ring network design problem. Each of these formulations represents a different tradeoff along the continuum between model detail and tractability. Like many approaches based on integer programming, however, there are inherent limitations in the scalability of these techniques to larger network designs. Nonetheless, these formulations can generate good solutions for relatively small networks and serve as useful benchmarks for other methods.

Chapter 11 developed a novel Tabu Search meta-heuristic for the multi-ring network design problem, which guides a local search procedure to explore regions in the solution space beyond a local optimum. Despite the relative simplicity of this procedure, it performed extremely well in all the test cases, providing solutions that were within 3% of the best solutions found by any method.

Chapter 12 compared the performance of the design methods and improvement heuristics studied in Chapters 9 through 11. The two main performance criteria considered were solution quality and runtime. The results showed that the Tabu Search meta-heuristic consistently provided the best or second best solutions in all test cases. Overall, improvements of up to 32% relative to the benchmark solution were realized using the design methods proposed and implemented here. Lower bounding procedures and statistical inference were also used for the first time on the multi-ring network design problem to quantify solution quality. These results suggest that the best solutions for each test case lie within about 12% of the estimated optimal solution.

13.3 Summary of Main Contributions

The major original contributions established by this work include the following:

1. Engineering design, development and testing of a software architecture, data model and algorithms for a comprehensive planning tool and experimental platform for ring-based transport network design (RingBuilder Interactive 1.0).
2. First formal presentation and development of the ring loading problem as distinct from the ring sizing problem.
3. First mathematical programming formulation of the ring loading problem and systematic study of demand splitting and time-slot interchange on the loading efficiency in bidirectional line-switched rings.
4. The conception, development and implementation of a demand packing algorithm for ring-based transport networks.
5. Origination of three mathematical programming formulations of the multi-ring network design problem.
6. The conception and development of two lower bounding procedures for the multi-ring network design problem.
7. Conception, development and implementation of a Tabu Search meta-heuristic to the multi-ring network design problem.
8. Development and implementation of an adaptive dithered sequencing meta-heuristic.
9. Development and implementation of a new balanced ring loading algorithm.
10. Systematic study of all design methods and improvement heuristics developed in items 4 through 9 above.
11. Origination and implementation of an improved algorithm for finding all cycles in an undirected graph.

13.3.1 Publications

Several papers and technical reports based on this work have either been published or are currently in review. These include the following:

1. G.D. Morley and W.D. Grover, "Loading efficiency of bidirectional rings under different routing and technology constraints," *Journal of Network and Systems Management*, accepted for publication.
2. G.D. Morley and W.D. Grover, "Comparison of Mathematical Programming Approaches to Optical Ring Design," *Proc. Canadian Conf. on Broadband Research*, November 1999, pp. 173-184.
3. G.D.Morley and W.D. Grover, "Current Approaches in the Design of Ring-based Optical Networks," *Proc. IEEE Canadian Conf. on Elec. and Comp. Eng. '99*, May 1999, pp. 135-144.
4. G.D. Morley and W.D. Grover, "Optimal Loading of SONET BLSRs: Assessment of Benefits of Demand Splitting and Time-Slot Interchange," *Proc. Canadian Conf. on Broadband Research*, 1998, pp.135-144.
5. G.D. Morley and W.D. Grover, "A Comparative Survey of Methods for Automated Design of Ring-based Transport Networks," *Technical Report TR-97-04*, TRLabs, 1998.
6. C.Y. Lee, W.D. Grover and G.D. Morley, "Heuristic Methods for the 'Span Elimination' Problem in Ring-Based Transport Network Design," *Proc. IEEE Canadian Conf. on Elec. and Comp. Eng. '99*, May 1999, pp. 232-237.
7. G.D. Morley and W.D. Grover, "RingBuilder SHR Design Study: SaskTel Metropolitan Regina Network," *Technical Report TR-97-03(R)*, TRLabs, 1998.
8. M. Jeremiah, G.D. Morley, *RingBuilder Interactive Users Guide*, TRLabs, March 2000.

A patent application covering several aspects of the work presented in this thesis was also filed in May 2000 by TR*Labs*. It is also worth noting that the software tools developed as part of this thesis have also been licensed from TR*Labs* by Virtual Photonics Inc. for use in their commercial transport network planning software. In addition, they have been transferred in various forms to Nortel Networks, Telus, Sasktel, MCIWorldcom and TelOptica.

13.4 Topics for Further Research

13.4.1 Advanced Tabu Search Procedures

Several ideas for enhancing the Tabu Search meta-heuristic were identified during the course of this work. One would be to randomize move selections at each iteration to counteract the uncertainty (or noise) inherent in the current move evaluation process. That is, the transport efficiency is not always an accurate predictor of move quality and the search procedure may benefit from some form of move randomization. This technique has been successfully used in other applications of Tabu Search. One alternative is to use a modified version of the probabilistic selection technique developed in Chapter 8.

A detailed analysis of the search trajectories also revealed that the majority of the runtime is spent evaluating constructive (add) moves. Therefore, substantial improvements in runtime could be realized if the procedure for evaluating add moves were to be improved. One approach would be to consider only a subset of ring candidates based on the intersection with unserved demands and/or possibly the existing ring set. This could also be combined with the probabilistic move selection procedure described above.

The search results also show that in some cases certain rings are frequently swapped into and out of the solution. This usually occurs in regions with limited connectivity where only a handful of candidate rings may be able to serve the demand. As a result, the search may exhibit a tendency to cycle through the available candidates, particularly when the utilization of the rings is low. This suggest that improvements could be obtained by taking these factors into account during move evaluation.

Consideration should also be given to incorporating other types of moves to allow the candidate ring set to evolve over time. This would be particularly beneficial in large design problems where only a small fraction of the cycle set can be considered due to memory and runtime constraints. Additional moves could include, for example, the merge, split and enlargement operations used in the local search procedure by Gardner et al. [GST95].

Several other refinements of the Tabu Search can be envisaged. For example, varying the drop depth during the search procedure could be used to alternate between periods of search intensification and diversification. That is, lower drop depths could be used to effect greater changes in the solution from one iteration to the next. An alternative strategy would be to vary the tabu tenure of moves based on their influence on the solution quality. This technique has been applied with some success to other problems.

13.4.2 Comparison of Demand Packing Algorithm

It would be of some interest to compare the quality of solutions generated with the heuristic demand packing algorithm developed herein with the globally optimal solution. This would involve formulating the problem as an IP and solving several small test cases using the CPLEX MIP Solver, for example. Depending on the runtime results, more sophisticated techniques such as column generation may be required to find the strictly optimal solution. Alternatively, the LP (or possibly Lagrangean) relaxation could also be used to establish lower bounds for larger problem instances. Although optimal solutions are not a critical requirement in practice, it is clearly desirable to know the quality of the solutions generated using the current demand packing algorithm to assess whether further work is warranted on this problem.

13.4.3 Comparison of Balanced Ring Loading Algorithm

Similarly, it would interesting to compare the quality of solutions generated with the balanced ring loading algorithm with optimal solutions from the IPs (or variants thereof) developed in Chapter 4. This would likely involve reformulating the IP in Chapter 4 to capture additional problem details such as ADM and inter-ring transition costs. Rather than solving explicitly for the location of ADMs within the ring, a program could be written to simply enumerate all possible combinations of ADM locations and then each subproblem could be solved independently using the CPLEX MIP Solver, for example. Once again, optimal solutions are not a critical requirement in practice but it would be useful to know the quality of the solutions generated using the current algorithm to assess whether further work is warranted.

13.4.4 Further Research on Topology Optimization

Several alternatives for topology optimization were also conceived during the course of this work. In particular, the lower bounding procedure and IP (i.e., LBIP) developed in Chapter 7 could be used directly as a replacement for the SCIP formulation in the span elimination work by Lee et al. [Lee00]. This would greatly improve the runtime performance of the algorithms developed by Lee with little or no impact on the solution quality. Heuristics based on the same conditioning arguments can also be contemplated. If the number of excess ring modules at each node is less than two, a solution can be found using the heuristic procedure described in Section 6.3.3. Heuristics similar to those developed for the multiple postman problem [Kno89] may also prove useful.

Another promising alternative would be to modify the lower bounding procedure with route optimization (i.e., LBRIP) to directly address the span elimination problem. One distinct advantage of this approach is that it not only determines which spans to eliminate but also optimizes the flow over the remaining spans to minimize the total transport capacity required. It would also be worth investigating whether column generation, which has been used extensively on other multi-commodity flow problems [Hu92], could be used to allow larger problem instances to be solved optimally. Once solved, this procedure could form the basis for an entirely new network design approach. This would involve developing an algorithm for assigning ring modules to individual rings, such that no ring has repeated spans or nodes. Several example designs have already been constructed by hand from the LBRIP results from Chapter 11 but further work is required to develop an algorithm for automating this process. These initial designs already show that, in general, several different designs can be generated from a single (LBRIP) solution. For example, two designs were generated based on the LBRIP solutions for test case 2 (Net15, 4B48). While the total fibre mileage for both designs was the same, the number of inter-ring transitions was roughly five times higher in one of the designs. This suggests that a second level of optimization would be useful in constructing a ring set (i.e., design) from the initial solution.

13.4.5 Multi-period Planning Enhancements

A naive approach to the multi-period planning problem involves generating independent designs for each time period (starting from the most current time period) and packing as much demand as possible into the existing rings between periods. Clearly, this approach is unlikely to find the optimal solution.

Although not yet implemented or tested, a simple but intuitively attractive alternative was conceived during the course of the work. This approach involves fairly minor changes to the current ring loading and selection process in the current greedy heuristic algorithm. The basic idea is to modify the transport efficiency metric of candidate rings to discount the value of any future demands loaded onto a ring. This basic procedure works as follows:

At each iteration, candidate ring loading is performed in the usual manner except that the demands are first sorted in order of the time period in which they originate. That is, demands in the current period are loaded first, followed by demands in the second period and so on. After all demands have been loaded, the transport efficiency metric is calculated by discounting the value of demands loaded in future time periods. This is based on the notion that the revenue from future demands must be discounted (in the economic sense) to reflect their present value. It also recognizes the fact that the uncertainty surrounding a demand forecast increases with its distance into the future. Put simply, a demand today is worth more than a demand tomorrow. Thus, a ring candidate that carries demands originating exclusively in the current time period will have a higher transport efficiency than one that carries demands originating in future periods, all other things being equal.

After all candidate rings have been evaluated, the candidate ring with the highest transport efficiency is selected, but only the demands originating in the current period are actually loaded onto the ring. This process continues until all demands in the current period are served. Then the demand packing algorithm is used to pack demands from the next period into the rings selected in the current period. The entire process repeats until the demand in all time periods has been served.

Clearly, this approach is still in the early stages of development and requires further work to fully implement and test the idea.

13.4.6 Sensitivity Analysis of Design Results

An underlying assumption of the design methods developed in this thesis is that the decision environment is deterministic. That is, we assume that design inputs such as the demand pattern are known in advance or can be forecasted with little uncertainty. Experience suggests, however, that demand forecasts are not particularly accurate, especially in recent years. An appropriate question, therefore, is how sensitive are the network designs generated herein to variations in the demand pattern. A practical approach for addressing this question is to generate a number of demand patterns for each design and then use the demand packing algorithm developed in Chapter 8 to determine the percentage of demand served for each pattern. The demand patterns could be generated by randomly varying the demands in the orginal demand pattern according to some distribution. The percentage of demand served gives an indication of how "brittle" the design is to changes. The working load on each span could also be examined to determine which spans are most likely to be overloaded due to changes in the demand pattern. This information could be used to create more robust designs by reserving additional capacity on these spans.

Bibliography

[AKL84] A.D. Aleksandrov, A.N. Kolmogorov and M.A. Lavrent'ev, *Mathematics: Its Content, Methods and Meaning*, 2nd Ed., translated by K. Hirsch, M.I.T. Press, Cambridge, MA, 1984.

[ANS95a] ANSI, *Synchronous Optical Network (SONET) - Basic Description including Multiplex Structure, Rates and Formats*, ANSI T1.105-1995, 1995.

[ANS95b] ANSI, *Synchronous Optical Network (SONET) - Automatic Protection Switching*, ANSI T1.105.01-1995, 1995.

[Ant97] A. Antonopoulos, "Planning Telecommunication Transport Networks: Metrication, Analysis and Design," Ph.D. Thesis, University College London, 1997.

[Bat96] R. Battiti, "Reactive Search: Toward Self-tuning Heuristics," *Modern Heuristic Search Methods*, Editors V.J. Rayward-Smith et al., Wiley, 1996, pp. 61-83.

[BeG92] D. Bertsekas and R. Gallager, *Data Networks*, 2nd Ed., Prentice All, Englewood Cliffs, NJ, 1992, pp. 113.

[Bel88] Bellcore, *Digital Cross-Connect System (DCS) Requirements and Objectives*, TR-TSY-000170, Issue 1, November 1988.

[Bel93] Bellcore Special Report SR-NWT-002514, *Digital Cross-Connect Systems in Transport Network Survivability*, Issue 1, January 1993.

[Bel95a] Bellcore, *SONET Dual-Fed Unidirectional Path Switched Ring (UPSR) Equipment Generic Criteria*, GR-1400-Core, Issue 1, Revision 1, October 1995.

[Bel95b] Bellcore, *SONET Bidirectional Line-Switched Ring Equipment Generic Criteria*, GR-1230-Core, Issue 2, November 1995.

[BGS94] G.N. Brown, W.D. Grover, J.B. Slevinsky and M.H. MacGregor, "Mesh/Arc Networking: An Architecture for Efficient Survivable Self-Healing Networks," *Proc. of the 1994 IEEE Intl. Conf. on Communications, Supercomm/ICC '94*, Vol. 1, May 1994, pp. 471-477.

[BoM76] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications*, North-Holland, New York, 1976.

[BTR96] S. Bortolon, H.M.F. Tavares, R.V. Ribiero, E. Quaglia and A. Bergamaschi, "A Methodology To SDH Network Design Using Optimization Tools," *Proc. of IEEE Globecom '96*, 1996, pp. 1867-1871.

[CCR99] F.Callegati, M. Casoni, C. Rafaelli,and B. Bostica, "Packet Optical Networks for High-Speed TCP-IP Backbones," *IEEE Communications Magazine*, Vol. 37, No. 1, 1999, pp. 124-129.

[CDS95] S. Cosares, D.N. Deutsch, I. Saniee and O. Wasem, "SONET Toolkit: A Decision Support System for Designing Robust and Cost-Effective Fiber-Optic Networks," *Interfaces*, Vol. 25, 1995, pp. 20-40.

[Che91] P.-S. Cherng, "Comparison of Bidirectional and Unidirectional SDH Rings," *European Telecommunications Standards Institute*, ETSI TM3/WT22, April 1991.

[ChO99] T.M. Chen and T.H. Oh, "Reliable Services in MPLS," *IEEE Communications Magazine*, Vol. 37, No. 12, 1999, pp. 58-62.

[ChS97] S. Chamberland and B. Sansó, "Heuristics for Ring Network Design when Several

Types of Switches are Available," in *Proc. ICC '97*, Vol. 2, 1997, pp. 570-574.

[Cis99] Cisco Systems, *Dynamic Packet Transport Solution*, GSR 12000 OC-12c/STM-4c Packet Ring Line Card, Data Sheet, San Jose, CA, 1999.

[CKY96] S.-H. Chung, H.-G. Kim, Y.-S. Yoon and D.-W. Tcha, "Cost-Minimizing Construction of a Unidirectional SHR with Diverse Protection," *IEEE/ACM Trans. Networking*, Vol. 4, No. 6, Dec. 1996, pp. 921-928.

[Com97] Compass Modeling Solutions, *Using AMPL*, 1997.

[CoS94] S. Cosares and I.Saniee, "An optimization problem related to balancing loads on SONET rings," *Telecommunications Systems*, No. 3, 1994, J.C. Baltzer AG, Science Publishers, pp. 165-181.

[CPL98] CPLEX Optimization Inc., *Using the CPLEX Callable Library*, Version 6.0, 1998.

[CQT96] L.A. Cox Jr., Y. Qui, G.E. Tegan and L. Lu, Method and System for Planning and Installing Communication Networks," U.S. Patent No. 5,515,367, May 7, 1996.

[CSW92] S. Cosares, I. Saniee and O. Wasem, "Network Planning with the SONET Toolkit," *Bellcore Exchange*, September/October 1992, pp. 8-13.

[DDH95] B.T. Doshi, S. Dravida and P. Harshavardhana, "Overview of INDT-A New Tool for Next Generation Network Design," *Proc. IEEE Globecom*, Singapore, November 1995.

[DDH97] B.T. Doshi, S. Dravida, P. Harshavardhana, P.K. Johri, and R. Nagarajan, "Dual (SONET) Ring Interworking: High Penalty Cases And How To Avoid Them," *Proc. of ITC 15*, Elsevier Science, 1997, pp. 361-370.

[Dem99] P. Demeester et al., "Resilence in Multilayer Networks," *IEEE Communications Magazine*, Vol. 37, No. 8, 1999, pp. 70-75.

[DGM94] D.A. Dunn, W.D. Grover and M.H. MacGregor, "Comparison of k-Shortest Paths and Maximum Flow Routing for Network Facilities Restoration," *IEEE J. on Selected Areas in Communications*, Vol. 12, No. 1, 1994, pp. 88-99.

[Dij59] E.W. Dijkstra, "A note on two problems in connection with graphs," *Number Math.*, Vol. 1, 1959, pp. 269-271.

[Dos97] B.T. Doshi et al., "Integrated Network Design Tools (INDT): A Suite of Network Design Tools for Current and Next Generation Networking Technologies," *Proc. of IEEE ISSC '97*, Alexandria, Egypt, 1997, pp. 332-338.

[DPW99] R.D. Doverspike, S. Phillips and J.R. Westbrook, "Future Transport Network Architectures," *IEEE Communications Magazine*, Vol. 37, No. 8, 1999, pp. 96-101.

[DWY99] P. Demeester, T.-H. Wu and N. Yoshikai, "Survivable Communication Networks," *IEEE Communications Magazine*, August 1999, pp. 40-42.

[EBC98] G. Ellinas, K. Bala and G.-K. Chang, "A Novel Wavelength Assignment Algorithm for 4-fiber WDM Self-Healing Rings," in: *Proc. of ICC '98*, 1998, pp. 197-201.

[EdJ73] J. Edmonds and E.L. Johnson, "Matching, Euler Tours and the Chinese Postman Problem," *Mathematical Programming*, Vol. 5, 1973, pp. 88-124.

[Fal90] W.E. Falconer, "Service Assurance in Modern Telecommunication Networks," *IEEE Communications Magazine*, June 1990, pp. 32-39.

[FiT28] R. Fisher and L. Tippett, "Limiting forms of the frequency distribution of the largest or smallest member of a sample," *Proc. Camb. Phil. Soc.*, 1928, Vol. 24, pp. 180-190.

[Fla90] T. Flanagan, "Fiber Network Survivability," *IEEE Communications Magazine*, June 1990.

[FSV98] A. Fink, G. Schneidereit and S. Voβ, "Ring network design for metropolitan area networks," Technical Report, Technical University of Braunschweig, 1998, pp. 1-14.

[FSV99] A. Fink, G. Schneidereit and S. Voß, "Solving General Ring Network Design Problems by Meta-Heuristics", Computing Tools for Modeling, *Optimization and Simulation (Interfaces in Computer Science and Operations Research)*, M. Laguna, J.L. González Velarde (Eds.), Kluwer, Boston, 1999, 91–113.

[GaJ79] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, 1979.

[GCF99] B. Gendron, T.G. Crainic and A. Frangioni, "Multicommodity Capacitated Network Design," *Telecommunications Network Planning*, Editors B. Sansò and P. Soriano, Kluwer Academic Publishers, 1999, pp. 1-19.

[GeK97] O. Gerstel and S. Kutten, "Dynamic Wavelength Allocation in All-Optical Ring Networks," in: *Proc. of ICC '97*, 1997, pp. 432-436.

[GHS94] L.M. Gardner, M. Heydari, J. Shah, I.H. Sudborough, I.G. Tollis, and C. Xia, "Techniques for Finding Ring Covers in Survivable Networks," *Proc. of IEEE Globecom '94*, 1994, pp. 1862-1866.

[GJM80] M.R. Garey, D.S. Johnson, G.L Miller and C.H. Papadimitriou, The Complexity of Coloring Circular Arcs and Chords, *SIAM J. Alg. Disc. Math.*, Vol. 1, No. 2 ,1980, pp. 216-227.

[GlL97] F. Glover and M Laguna, *Tabu Search*, Kluwer Academic Press, 1997.

[GLL95] M. Gendreau, M. Labbe and G. Laporte, "Efficient heuristics for the design of ring networks," *Telecommunication Systems*, J.C. Baltzer AG, Vol. 4, 1995, pp. 177-188.

[GLO98] O. Goldschmidt, A. Laugier and E.V. Olinick, "SONET/SDH Ring Assignment with Capacity Constraints," INFORMS Meeting, Montreal, Canada, April 26-29, 1998.

[GLS98] O. Gerstel, P. Lin, and G. Sasaki, "Wavelength Assignment In A WDM Ring To Minimize Cost Of Embedded SONET Rings," in: *Proc. of IEEE Infocom '98*, 1998, Vol. 1, pp. 94-101.

[GoA79] B.L. Golden and F.B. Alt, "Interval Estimation of A Global Optimum for Large Scale Optimization," *Naval Research Logistics Quarterly*, No. 26, 1979, pp. 69-77.

[GOP98] A.V. Goldberg, J.D. Oldham, S. Plotkin and C. Stein, "An Implementation of a Combinatorial Approximation Algorithm for Minimum-Cost Multicommodity Flow," *Integer Programming and Combinatorial Optimization, 6th International IPCO Conf.*, June 1998, Houston, Texas, pp. 338-352.

[Gro92] W.D. Grover, "Case Studies of Survivable Ring, Mesh and Mesh-Arc Hybrid Networks," in *Proc. IEEE Globecom '92*, Vol. 1, pp. 633-638.

[Gro94] W.D. Grover, "Distributed Restoration of the Transport Network," *Telecommunications Network Management into the 21st Century - Techniques, Standards, Technologies and Applications*, Editors S. Aidarous, T. Plevyak, IEEE, New York, NY, 1994, pp. 337-417.

[Gro96] W.D. Grover, "Analysis of Unavailability and Resource Consumption for High Availability Paths in SONET Ring-based Networks," *TRLabs Technical Report TR-96-01*, February 1996.

[Gro97a] W.D. Grover, "Matched Nodes and Dual Feeding: Options for High Availability Path Provisioning in SONET Ring-Based Networks," *Proc. of Canadian Conference on Broadband Research*, Ottawa, Canda, April 16-17, 1997, pp. 160-171.

[Gro97b] W.D. Grover, "Discussion Paper: Methods for Design of Cost Effective Networks of Self-Healing Rings," *TRLabs Internal Discussion Paper*, Jan. 23, 1997, pp. 1-14.

[Gro97c] W.D. Grover, "Network Survivability: A Crucial Issue for the Information Society," *IEEE Canadian Review*, No. 27, 1997, pp. 16-21.

[Gro99] W.D. Grover, "High Availability Path Design in Ring-Based Optical Networks," *IEEE/ACM Trans. Networking*, Vol. 7, No. 4, 1999, pp. 558-574.

[GSM95] W.D. Grover, J.B. Slevinsky, M.H. MacGregor, "Optimized design of ring-based survivable networks," *Canadian J. Elect. & Comp. Eng.*, Vol. 20, No. 3, 1995.

[GST95] L.M. Gardner, I.H. Sudborough, I.G. Tollis, "Netsolver: A Software Tool For the Design of Survivable Networks," *Proc. of IEEE Globecom '95*, 1995, pp. 926-930.

[HeS82] D.P. Heyman and M.J. Sobel, *Stochastic Models in Operations Research*, McGraw-Hill, New York, NY, 1982.

[HJN96] P. Harshavardhana, P.K. Johri and R. Nagarajan, "A note on weight-based load balancing on SONET rings," *Telecommunication Systems*, Vol. 6, 1996, pp. 237-239.

[HoF91] C.T. Horngren and G. Foster, *Cost Accounting: A Managerial Emphasis*, 7th ed., Prentice Hall, Englewood Cliffs, NJ, 1991, pp. 29-34

[Hu92] T.C. Hu, *Combinatorial Algorithms*, Addison-Wesley, Reading, MA, 1992, pp. 40-83.

[ITU93] ITU-T, *Network Node interfaces for the Synchronous Digital Hierarchy (SDH)*, ITU-T Recommendation G.707, 1993.

[ITU95] ITU-T, *Transmission and Multiplexing (TM): Generic Functional Architecture of Transport Networks*, ITU-T Recommendation G.805, 1995.

[ITU96] ITU-T, *Types and Characteristics of SDH Network Protection Architectures*, ITU-T Recommendation G.841, 1996.

[JHV99] D. Johnson, N. Hayman and P. Veitch, "The Evolution Of A Reliable Transport Network," *IEEE Communications Magazine*, Vol. 37, No. 8, 1999, pp. 52-57.

[Joh75] D.B. Johnson, "Finding All the Elementary Circuits of a Directed Graph," *SIAM J. on Computing*, Vol. 4, 1975, pp. 77-84.

[Joy00] B. Joy et al., *The Java Language Specification, Second Edition*, Addison-Wesley, 2000.

[KaC94] N. Karunanithi, T. Carpenter, "A Ring Loading Application of Genetic Algorithms," *Proc. of the ACM Symposium on Applied Computing*, 1994, pp. 227-231.

[KaC97] N. Karunanithi and T. Carpenter, SONET ring sizing with genetic algorithms, *Computers and Operations Research*, Vol. 24, No. 6, 1997, 581-591.

[Kam93] I.P. Kaminow, "Photonic Networks," *The Electrical Engineering Handbook*, R.C. Dorf, Editior-in-Chief, CRC Press, 1993, pp. 1439.

[KaO99] R. Kawamura and H. Ohta, "Archtectures for ATM Network Survivability and Their

Field Deployment," *IEEE Communications Magazine*, Vol. 37, No. 8, 1999, pp. 88-94.

[Kar86] R.M. Karp, "Combinatorics, Complexity, and Randomness," *Communications of the ACM*, Vol. 29, No. 2, February 1986, pp. 98-117.

[Kha97] S. Khanna, "A Polynomial Time Approximation Scheme for the SONET Ring Loading Problem," *Bell Labs Technical Journal*, 1997, pp. 36-41.

[KKG91] A. Kershenbaum, P. Kermani, G. Grover, "MENTOR: An Algorithm for Mesh Network Topological Optimization and Routing," *IEEE Trans. on Comm.*, No. 4, 1991, pp. 503 - 513.

[Kno89] T.W. Knowles, *Management Science: Building and Using Models*, Irwin, Homewood IL, 1989.

[KNR97] J.L. Kennington, V.S.S. Nair, M.H. Rahman, "Optimization Based Algorithms for Finding Minimal Cost Ring Covers in Survivable Networks," *Technical Report 97-CSE -12*, Southern Methodist University, Dallas, Texas, August 1997, pp. 1-26.

[Law82] J.F. Lawless, *Statistical Models and Methods for Lifetime Data*, Wiley, 1992.

[LeC97] C.Y. Lee and S.G. Chang, Balancing loads on SONET rings with integer demand splitting, *Computers and Operations Research*, Vol. 24, No. 3, 1997, pp. 221-229.

[Lee00] C.Y. Lee, "Heuristic Methods for Sub-Graph Topology Enhancement in Ring-Based Transport Network Design," M.Sc. Thesis, University of Alberta, 2000.

[LGM99] C.Y. Lee, W.D. Grover and G.D. Morley, "Heuristic Methods for the 'Span Elimination' Problem in Ring-Based Transport Network Design," *Proc. IEEE Canadian Conf. on Elec. and Comp. Eng. '99*, May 1999, pp. 232-237.

[LRT93] C.-H. Lee, H.-B. Ro and D.-W. Tcha, "Topological Design of a Two-Level Network with Ring-Star Configuration," *Computer Ops. Res.*, Vol. 20, No. 6, 1993, pp. 625-637.

[Mac91] M. MacGregor, "The Self Traffic-Engineering Network," Ph.D. Thesis, University of Alberta, 1991.

[MaD76] P. Mateti and N. Deo, "On algorithms for enumerating all circuits of a graph," *SIAM J. Comput.*, vol. 5, no. 1, Mar. 1976, pp. 90-99.

[MaG93] M. MacGregor, W.D. Grover, "Optimized *k*-shortest paths algorithm for facility restoration," *Software Practice and Experience*, 1993.

[MBN99] J. Manchester, P. Bonenfant and C. Newton, "The Evolution Of Transport Network Survivability," *IEEE Communications Magazine*, Vol. 37, No. 8, 1999, pp. 44-51.

[McB98] A. McGuire and P. Bonenfant, "Standards: The Blueprint for Optical Networking," *IEEE Communications Magazine*, Vol. 32, No. 2, 1998, pp. 68-78.

[MKT97] Y.-S. Myung, H.-G. Kim and D.-W. Tcha, Optimal Load Balancing on SONET Bidirectional Rings, *Operations Research*, Vol. 45, No. 1, 1997, 148-152.

[Min91] D. Minoli, *Telecommunications Technology Handbook*, Artech House Inc., Boston, Mass., 1991.

[MoG98a] G.D. Morley, W.D. Grover, "Optimal Loading of SONET BLSRs: Assessment of Benefits of Demand Splitting and Time-slot Interchange," *Proc. 2nd Canadian Conference on Broadband Research*, Ottawa, Canada, June 22-24, 1998, pp. 135-144.

[MoG98b] G.D. Morley and W.D. Grover, "A Comparative Survey of Methods for Automated Design of Ring-based Transport Networks," Technical Report TR-97-04, TRLabs, 1998.

[MoG99a] G.D. Morley and W.D. Grover, "Comparison of Mathematical Programming Approaches to Optical Ring Design," *Proc. Canadian Conf. on Broadband Research*, November 1999, pp. 173-184.

[MoG99b] G.D.Morley and W.D. Grover, "Current Approaches in the Design of Ring-based Optical Networks," *Proc. IEEE Canadian Conf. on Elec. and Comp. Eng. '99*, May 1999, pp. 135-144.

[Moy98] J.T. Moy, *OSPF: Anatomy of an Internet Protocol*, Addison-Wesley, 1998.

[Mur92a] K. Murty, *Network Programming*, Prentice-Hall, Englewood Cliffs, NJ, 1992, pp. 365-368.

[Mur92b] K. Murty, *Network Programming*, Prentice-Hall, Englewood Cliffs, NJ, 1992, pp. 209-222.

[NeW99] G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, 1999, pp. 540-546.

[Nor94] Northern Telecom, *SONET 101: An Introduction to Basic Synchronous Optical Networks*, Northern Telecom, 56118.11, Issue 3, October 1994.

[Nor96] Northern Telecom, *Introduction to SONET Networking*, 1996.

[OwW93] H.L. Owen, C. Wulf-Mathies, "Synchronous Digital Hierarchy Optical Metropolitan Network Ring Traffic Analysis," *European Trans. Telecomm.*, Vol. 6, No. 4, 1993, pp. 591-597.

[Owe96] H.L. Owen, "Ring-based bandwidth dimensioning in SDH networks," submitted to *Computer Communications*, Elsevier.

[RaS95] R. Ramaswami and K.N. Sivarajan, "Routing and Wavelength Assignment in All-Optical Networks," *IEEE/ACM Trans. on Networking*, Vol. 3, 1995, pp. 489-500.

[RaS97] R. Ramaswami and G. H. Sasaki. "Multiwavelength optical networks with limited wavelength conversion," in: *Proc. of IEEE Infocom '97*, 1997, Vol. 2, pp. 489-498.

[RaS98] R. Ramaswami and K.N. Sivarajan, *Optical Networks: A Practical Perspective*, Academic Press, 1998.

[Ree93] C.R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*, John Wiley and Sons, New York, 1993.

[RND77] E.M. Reingold, J. Nievergelt, N. Deo, *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall, Englewoood Cliffs, NJ, 1977.

[Rob94a] A. Roberts, "Graduate seminar on algorithms for survivable network design employing self healing rings," Dept. of Electrical Engineering, U. of Colorado, Colorado Springs, Colo., June 10, 1994.

[Rob94b] A. Roberts, A. DeGuilio, "A Network Design Methodology," CS622 Project Report, Dept. of Electrical Engineering, U. of Colorado, Colorado Springs, Colo., 1994.

[Rya98] J.P. Ryan, "WDM: North American Deployment Trends," *IEEE Communications Magazine*, Vol. 36, No. 2, 1998, pp. 40-44.

[SGM93] J.B. Slevinsky, W.D. Grover and M.H. MacGregor, "An algorithm for survivable network design employing multiple self-healing rings," in *Proc. IEEE Globecom '93*, Dec. 1993, pp. 1568-1573.

[Sha98] C.A. Shaffer, *A Practical Introduction to Data Structures and Algorithm Analysis, Java*

Edition, Prentice Hall, Upper Saddle River, NJ, 1998.

[ShF94] J. Shi and J. Fonseka, "Design of Hierarchical Self-Healing Ring Networks," *Proc. of IEEE Intl. Conf. on Communications '94*, 1994, pp. 478-482.

[ShF96] J. Shi and J. Fonseka, "Interconnection of Self-Healing Rings," *Proc. of IEEE Intl. Conf. on Communications '96*, 1996, pp. 1563-1567.

[Shi95] J. Shi, "Design and Analysis of Survivable Telecommunications Networks," Ph.D. Thesis, University of Texas at Dallas, 1995.

[SHS93] M.M. Slominski, S. Hasegawa, H. Sakauchi, H. Okazaki, "Multi-ring Approach for ATM-VP Networks Survivability," *Proc. of IEEE Intl. Conf. on Communications '93*, 1993, pp. 245-249.

[Sle99] J.B. Slevinsky, "Synthesis of Ring-based Restorable Networks," M.Sc. Thesis, University of Alberta, 1999.

[SSW98] A. Schrijver, P. Seymour and P. Winkler, "The Ring Loading Problem," *SIAM J. Discrete Math.*, Vol. 11, No. 1, February 1998, pp. 1-14.

[Sta99] *StatSoft Inc.*, Electronic Statistics Textbook, Tulsa, OK, 1999, WEB: http://www.statsoft.com/textbook/stathome.html.

[StG00] D. Stamatelakis, W.D. Grover, "Theoretical Underpinnings for the Efficiency of Restorable Networks Using Preconfigured Cycles ("p-cycles")," *IEEE Trans. Comm.*, Vol. 48, No. 8, pp. 1262-1265.

[SWS99] P. Soriano, C. Wynants, R. Séguin, M. Labbé, M Gendreau and B. Fortz, "Design and Dimensioning of Survivable SDH/SONET Networks," *Telecommunications Network Planning*, Editors B. Sansò and P. Soriano, Kluwer Academic Publishers, 1999, pp. 147-167.

[Sui99] E. Siu, "Methods and Issues in Path Provisioning on Ring-based Networks," M.Sc. Thesis, University of Alberta, Fall 1999

[TRL00a] TRLabs, *RingBuilder Interactive Application Programmers Interface*, March 2000.

[TRL00b] TRLabs, *RingBuilder Interactive Users Guide*, March 2000.

[Tuc75] A. Tucker, "Coloring a Family of Circular Arcs," *SIAM J. Appl. Math.*, Vol. 29, No. 3, 1975, pp. 493-502.

[VSK96] R. Vachani, A. Shulman, P. Kubat, "Multicommodity Flows in Ring Networks," *INFORMS Journal on Computing*, Vol. 8, No. 3, Summer 1996, INFORMS, pp. 235-242.

[Was91a] O.J. Wasem, "An Algorithm for Designing Rings for Survivable Fiber Networks," *IEEE Trans. on Reliability*, Vol. 40, No. 4, October 1991, pp. 425-432.

[Was91b] O.J. Wasem, "Optimal Topologies for Survivable Fiber Optic Networks using SONET Self-Healing Rings," *Proc. of IEEE Globecom '91*, 1991, pp. 2032-2038.

[WCY99] E.W.M. Wong, A.K.M. Chan and T.-S.P. Yum, "A Taxonomy of Rerouting in Circuit-Switched Networks," *IEEE Communications Magazine*, Vol. 37, No. 11, 1999, pp. 116-122.

[Win94] W.L.Winston, *Operations Research: Applications and Algorithms*, 3rd Ed., Duxbury Press, Belmont CA, 1994.

[WKL92] T.-H. Wu, D. Kong, R.C. Lau, "A Broadband Virtual Path SONET/ATM Self-Healing Ring Architectures and Its Economic Feasibility Study," IEEE Journal on Selected Areas of Communications, Vol. 9, No. 9, December 1992, pp. 834-840.

[Wol98] L.A. Wolsey, *Integer Programming*, John Wiley & Sons, 1998.

[Wu92] T.-H. Wu, *Fiber Network Service Survivability*, Artech House Inc., Boston, Mass., 1992.

[Wu99] T.-H. Wu, "Emerging Technologies for Fiber Network Survivability," *IEEE Communications Magazine*, Vol. 33, No. 2, 1999, pp. 70-75.

[WuC91] T.-H. Wu, R.H. Cardwell, "A Multi-period Design Model for Survivable Network Architecture Selection for SONET Interoffice Networks," *IEEE Trans. on Reliability*, Vol. 40, No. 4, October 1991, pp. 417-427.

[WuL90] T.-H. Wu and R.C. Lau, A Class of Self-Healing Ring Architectures for SONET Network Applications, in: *Proc. of Globecom '90*, 1990, pp. 444-449.

[WWC94] O.J. Wasem, T.-H. Wu and R.H. Cardwell, "Survivable SONET Networks - Design Methodology," *IEEE Journal on Selected Areas in Communications*, Vol. 12, No. 1, January 1994, pp. 205-212.

[XCG99] Xu, Chui and Glover, "Optimizing a Ring-Based Private Line Telecommunications Network Using Tabu Search," *Management Science*, Vol. 45, No. 3, 1999, pp. 330-345.

Appendix A: Ring Loading Formulations

The IP formulations for the two remaining problem variants are given in this section.

A.1 IP3: Channel Interchange, with Demand Splitting

The IP formulation for ring loading with channel interchange and demand splitting can be written as follows.

IP3 Maximize:

$$\sum_{k \in D} (f_k^+ + f_k^-) \qquad (1)$$

Subject to:

$$\sum_{k \in K_1(l)} f_k^+ + \sum_{k \in K_2(l)} f_k^- \le c, \qquad l = 1, \dots, n \qquad (2)$$

$$f_k^+ + f_k^- = d_k \cdot X_k, \qquad \forall k \in D \qquad (3)$$

$$0 \le f_k^+ \le d_k, \qquad \text{integer, } \forall k \in D \qquad (4)$$

$$0 \le f_k^- \le d_k, \qquad \text{integer, } \forall k \in D \qquad (5)$$

$$X_k \in \{0, 1\}, \qquad \forall k \in D \qquad (6)$$

where f_k^+ is the number of units of demand bundle k routed in the clockwise direction, f_k^- is the number of units of demand bundle k routed in the counter-clockwise direction, and X_k is a binary decision variable that indicates whether demand bundle k is selected for loading onto the given ring or not.

The objective function (1) is to maximize the total demand carried in the clockwise and counter-clockwise directions. Constraint set (2) ensures that the portion of demand carried (in both directions) over each span does not exceed its capacity. Constraint set (3) ensures that the sum of clockwise and counter-clockwise flows for each demand k is either equal to the total demand d_k or zero. Constraint sets (4) and (5) ensure that the clockwise and counter-clockwise flows for each demand bundle k do not exceed its size d_k. Constraint set (6) asserts that the routing decision variables are binary.

A.2 IP4: Channel Assignment, no Demand Splitting

The IP formulation for ring loading with channel assignment but without demand splitting can be written as follows.

IP4 Maximize:

$$\sum_{k\in D}\sum_{t=1}^{c}(f_{kt}^{+}+f_{kt}^{-}) \tag{7}$$

Subject to:

$$\sum_{k\in K_1(l)} f_{kt}^{+} + \sum_{k\in K_2(l)} f_{kt}^{-} \le 1, \qquad l = 1, \ldots, n, t = 1, \ldots, c \tag{8}$$

$$\sum_{t=1}^{c} f_{kt}^{+} = d_k \cdot X_k^{+}, \qquad \forall k \in D \tag{9}$$

$$\sum_{t=1}^{c} f_{kt}^{-} = d_k \cdot X_k^{-}, \qquad \forall k \in D \tag{10}$$

$$X_k^{+} + X_k^{-} \le 1, \qquad \forall k \in D \tag{11}$$

$$f_{kt}^{-}, f_{kt}^{+}, X_k^{+}, X_k^{-} \in \{0, 1\}, \qquad \forall k \in D, t = 1, \ldots, c \tag{12}$$

where $f_{kt}^{+} = 1$ if one unit of demand bundle k is routed over channel t in the clockwise direction and 0 otherwise. Similarly, $f_{kt}^{-} = 1$ if one unit of demand bundle k is routed over channel t in the clockwise direction and 0 otherwise.

The IP formulation for channel assignment with no demand splitting is identical to IP2 in Chapter 4 except that constraint set (16) is replaced by constraint sets (9) and (10), which ensure that all demands in a demand bundle are routed in either the clockwise or counter-clockwise direction or not at all. Once again, the objective function (7) is to maximize the demand carried in both directions around the ring over all channel positions. Constraint set (8) ensures that each channel t carries at most one demand over each span. Constraint sets (11) and (12) assert that all variables are binary.

Appendix B: Optimal Ring Loading Algorithm for Hubbed Demand Patterns

In this appendix we present a ring loading algorithm for finding the optimal routing and channel assignment for a pure hubbed demand pattern. We show that the solutions produced by this algorithm serve the maximum possible demand with or without channel interchange. We begin by observing that the maximum flow from any node on a ring is $2 \cdot c$, i.e. the aggregate capacity of its two incident spans. Because all demands in a pure hubbed demand pattern originate (or terminate) at the hub node, the upper bound on demand served is also $2 \cdot c$. When demand bundles must be loaded in their entirety or not at all, as assumed in this work, the maximum demand that can be served is

$$\max_{D' \subseteq D} \{q(D') \mid q(D') \leq 2 \cdot c\} \tag{1}$$

where D' is a subset of the complete demand pattern D and $q(D) = \sum_{j=2}^{n} d_{1j}$. In other words, the maximum demand that can be served under any circumstances is the maximal subset of demand bundles that satisfies the maximum flow condition.

For any such subset, a routing and channel assignment solution can be constructed that serves the entire subset of demands using the procedure given below (see Figure B1).

1. Starting at the hub node $k = 1$, proceed in the clockwise direction from node k to node $k = k + 1$.

2. If the load on span 1, $w_1 = c$ go to step 3. Otherwise, route as much demand as possible in the clockwise direction from the hub node to node k. Let us denote this demand flow as f_{1k}^{+}, where $f_{1k}^{+} = max(d_{1k}, c - w_1)$. Get the next f_{1k}^{+} available channel positions on span 1 and assign them to demand d_{1k} on spans 1 through *k*. If all of the demand can be routed in the clockwise direction, return to step 1. Otherwise, mark the current route node as the *cut node* and route the residual demand, $f_{1k}^{-} = d_{1k} - f_{1k}^{+}$, in the counter-clockwise direction. Get the first f_{1k}^{-} available channel positions on span *n* and assign them to demand d_{1k} on spans *n* through *k+1*. Return to step 1.

3. Route as much demand as possible in the counter-clockwise direction, such that the load

on span n is less than or equal to its capacity, $w_n = \sum_{j=k_c}^{n} f_{1j}^- \le c$. Get the next f_{1k}^- available channel positions on span n and assign them to demand d_{1k} on spans n through $k+1$. Return to step 1.

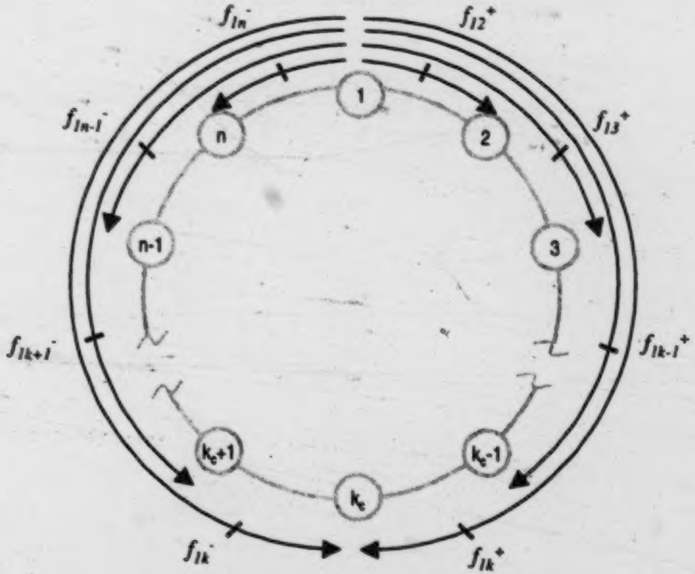


Figure B.1. Ring loading algorithm for pure hubbed demand pattern.

At the conclusion of this procedure, all demands have been served and each unit of demand is assigned a unique channel position along its entire route from the hub node to the destination node. To show that the entire demand is served, note that the demands on either side of the cut node, k_c, are routed entirely in one direction. That is, demands from the hub node to nodes $1 < k < k_c$ are routed in the clockwise direction (i.e., $f_{1k}^+ = d_{1k}$ and $f_{1k}^- = 0$) and demands from the hub node to nodes $k_c < k \le n$ are routed in the counter-clockwise direction (i.e., $f_{1k}^+ = 0$ and $f_{1k}^- = d_{1k}$). Demands from the hub node to the cut node k_c are split between the two directions. Thus the total demand flowing in the clockwise direction over span 1 is

$$w_1 = \sum_{j=2}^{k_c-1} f_{1j}^+ + f_{1k_c}^+ = \sum_{j=2}^{k_c-1} d_{1j} + f_{1k_c}^+ \le c \qquad (2)$$

and the demand flowing in the counter-clockwise direction over span n is

$$w_n = \sum_{j=k_c+1}^{n} f_{1j}^- + f_{1k_c}^- = \sum_{j=k_c+1}^{n} d_{1j} + f_{1k_c}^- \le c. \tag{3}$$

Adding Eq.(2) and Eq.(3) and rewriting using algebra yields

$$w_1 + w_n = \sum_{j=2}^{n} d_{1j} \tag{4}$$

Thus, it has been shown that the entire demand has been served. It is also clear that $w_i \le c$ for $1 < i < n$ and that the channels assigned to demands on each span are unique. For channel interchange, the only constraint is that the load on any span does not exceed its capacity. Because this has been shown to be true for channel assignment, the same routing is also valid for channel interchange.

Appendix C: Demand Matrices for Metropolitan Test Networks

The demand matrices for test networks Net15 and Net20 are listed in Figures C.1 and C.2, respectively.

To Node

From Node	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	3	2		1		1				1			1	2
2	-	4	3	1	1	2		1	1	1			2	3
3		-	2			1				1			1	
4			-		1	1	1	1	1	2			2	4
5				-					3	2				
6					-	2	2			13	1	1	2	4
7						-		2	2	8	1	1	2	4
8							-			6		4		
9								-	3	10	2	4	3	7
10									-	10	1	1	1	4
11										-	5	9	5	22
12											-	2		1
13												-		1
14													-	4

Figure C.1. Demand matrix for Net15 (in DS3s).

To Node

From Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1	1	1	4			1	1	1	1	1	1	15		1	1	1		11
1	-	1	1	2	1			1		1	1	4	7		1	1			1
2		-	1	1	2			1	2	1	1	1	8		1	1			1
3			-	2	1			1	2	3	1	1	13		1	1			1
4				-			1	1	1	1	1	1	13		1	1			
5					-			1	1	1	2	1	9		1	1			1
6						-													
7							-	1	1	1	1	2	8		1		1		
8								-	1	1	1	2	17		1	1	1	1	3
9									-	1	1	1	11		1		1		1
10										-	1	1	8		1	1	1		1
11											-	2	15		4	1	1	1	2
12												-	38		2	3	2	1	1
13													-		16	6	14	6	8
14														-					
15															-	1	1	1	1
16																-	1		1
17																	-		1
18																		-	1

Figure C.2. Demand matrix for Net20 (in DS3s).

Appendix D: Procedure for Estimating Optimal Solution Values

This appendix describes in further detail the procedure used to derive point estimates of the optimal solution. As noted in Chapter 7, the basic approach draws on the statistical theory of extreme values, which considers the case where n independent samples, each of size m, are taken from a population whose minimum value is γ. It has been shown that if the minimum value in sample i is v_i, the distribution of v_i approaches a 3-parameter Weibull distribution as $m \to \infty$ [FiT28]. The cumulative distribution function for the 3-parameter Weibull distribution is given by the expression:

$$F(x) = P(X \le x) = 1 - \exp\left\{-\left(\frac{x-\gamma}{\alpha}\right)^{\beta}\right\}, \qquad \alpha > 0, \beta > 0, \gamma < x \qquad (1)$$

where α is the scale parameter, β is the shape parameter and γ is the location parameter of the distribution.

For each test case, we estimate these parameters using the procedure described in [Sta99]. To illustrate this procedure we use the design results for test case 12 (Net43, 4B48 & 4B192). We begin by rank-ordering the solutions according to cost and computing the *median rank* for each solution using the following expression:

$$F(t) = (j - 0.3)/(n + 0.4) \qquad (2)$$

where $t = x - \gamma$, j denotes the solution order and n is the total number of solutions. The rank order, solution cost and median rank for all twenty-four solutions for test case 12 are shown in Table D.1.

Table D.1: Solution Data for Test Case 12

Rank, j	Solution Cost	Median Rank	Rank, j	Solution Cost	Median Rank
1	835.99	0.0287	13	923.15	0.5205
2	842.09	0.0697	14	928.67	0.5615
3	854.05	0.1107	15	934.69	0.6025
4	858.03	0.1516	16	941.33	0.6434
5	859.47	0.1926	17	944.75	0.6844
6	859.71	0.2336	18	945.93	0.7254
7	860.31	0.2746	19	949.25	0.7664
8	870.23	0.3156	20	953.81	0.8074
9	877.15	0.3566	21	962.51	0.8484
10	877.15	0.3975	22	967.33	0.8893
11	878.23	0.4385	23	985.97	0.9303
12	920.47	0.4795	24	1074.03	0.9713

Using this data, we can construct a probability plot, as shown in Figure D.1, for any value of the location parameter $0 \le \gamma < \min_{i} v_i$. Note that the horizontal axis is scaled logarithmically and the quantity $\ln(\ln(1/(1 - F(t))))$ is plotted about the vertical axis.

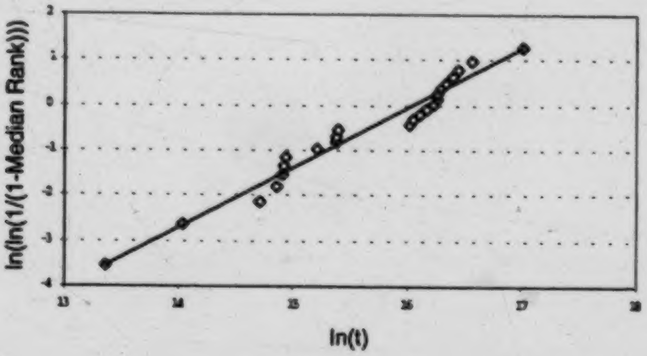


Figure D.1. Probability plot for test case 12 (Net43, 4B48 & 4B192).

The Weibull scale α and shape β parameters can be estimated from this plot by fitting a regression line to the empirical data. This is because the Weibull cumulative distribution can be rewritten in the form of a straight line $y = m \cdot x' + c$, as follows:

$$\ln(\ln(1/(1 - F(t)))) = \beta \cdot \ln(t) - \beta \cdot \ln\alpha \tag{3}$$

Thus, the shape parameter, β, is equal to the slope, m, of the regression line and the scale parameter, α, can be calculated as follows:

$$\alpha = e^{-c/\beta} \tag{4}$$

The quality of the fit between the regression line and the empirical data in the probability plot can be expressed by the common R^2 measure (correlation squared). Therefore, an estimate of the optimal solution value can be obtained by finding the value of the location parameter γ that maximizes R^2. This is illustrated in Figure D.2, which plots the correlation squared for several values of the location parameter. This plot shows that the best fit between the regression line and the empirical data occurs when location parameter is roughly equal to 830. Consequently, this value is taken as an estimate of optimal solution value for test case 12.

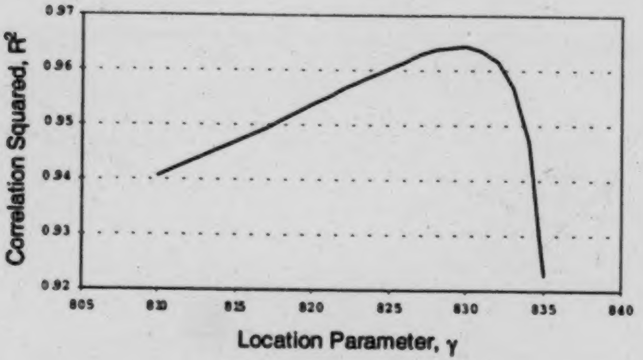


Figure D.2. Correlation squared, R^2, vs. location parameter.

Rather than constructing similar plots for each test case, the "Goal Seek" function in Microsoft Excel™ was used to find the best estimate of the location parameter.

Appendix E: Cycle Finding Algorithm

This appendix describes in detail the cycle finding algorithm mentioned in Chapter 8. The cycle finding algorithm developed here uses a depth-first search to enumerate all cycles in an undirected graph. For simplicity, we begin by describing the basic algorithm and then elaborate on the changes made to improve its efficiency. The algorithm is based on a similar algorithm by Johnson [Jon75] for directed graphs. Here, cycles are generated by adding edges to a path until either a cycle is found or the cycle circumference limits (in hops or distance) have been exceeded. To avoid exploring more paths than absolutely necessary, careful pruning is done to ensure that every cycle is generated only once.

The search begins at an arbitrary vertex s and a path $(s, v_1, v_2, \ldots, v_k)$ is built using a depth-first search. To ensure that each cycle is unique and not simply a permutation of a cycle found previously, we only consider cycles rooted at s. As vertices are added to the path, they are marked to indicate that they are unavailable for extending the current path. The search continues adding edges to the path until one of three conditions is met: (1) the path is blocked by a marked vertex $v_{k+1} \neq s$, (2) the cycle circumference limits are exceeded or (3) a cycle is found (i.e., $v_{k+1} = s$). After all edges out of v_k have been explored, we back up to the previous vertex. In doing so, we unmark v_k only if a cycle was found or the cycle circumference constraints were exceeded. Otherwise, if all paths extending out of v_k are blocked, it remains marked (i.e., unavailable) even after we back up past it. This prevents the algorithm from searching parts of the graph where previous searches have been unsuccessful. For each vertex w, a record is also kept of the adjacent vertices that are not in the current path but remain blocked by w. This set $B(w)$ of adjacent vertices is referred to as the *predecessors* of w. If w is unmarked when backing up to the previous vertex, then all of its predecessors are also unmarked using a recursive call. The predecessors in turn unmark all of their predecessors and so on. This makes w and its predecessors available again for extending the path.

The above process continues until we back up to the root vertex s. At this point, all of the cycles traversing the first edge (s, v_1) have been found so it is removed from the graph. This prevents duplicate cycles from being generated in the opposite direction. Once all edges incident on the root vertex have been explored, every cycle containing the root vertex will have been found. Note that it is not necessary to explore the last incident edge because all other edges have been removed and a cycle cannot be formed. Furthermore, all cycles containing the last edge will have already been found. The pseudo-code for the basic cycle finding procedure is given in Figure E.1.

```
procedure boolean Cycle(s,v) {
    flag = false
    avail(v) = false
    for each vertex w adjacent to v {
           if (w == s) {
                  output cycle = path + w
                  flag = true;
           } else if (path + w exceeds circumference constraints) {
                  flag = true;
           } else if (avail(w) == true) {
                  path = path + w
                  flag = flag OR Cycle(s,w)
           }
    }
    if (flag == true) {
           Unmark(v)
    } else {
           for vertex w adjacent to v {
                  B(w) = B(w) + v
           }
    }
    path = path - v
    return flag
}

procedure void Unmark(v) {
    avail(v) = true
    for each vertex v in B(w) {
           Unmark(v)
    }
}
```

Figure E.1 The recursive depth-first search procedure for finding cycles.

After all cycles containing the root vertex *s* have been found, the vertex is removed from the graph and the process begins again from the next vertex in the graph and so on. To complete the algorithm we observe that every cycle must lie within a two-vertex connected (or biconnected) component of the graph. Therefore, after the root vertex *s* is removed from the graph, we find the biconnected components of the subgraph and then call the cycle finding procedure for each component. This is also done recursively until all vertices have been explored. The pseudo-code for the main algorithm is given in Figure E.2.

```
procedure void FindCycles(G) {
    for each vertex v adjacent to s in G {
          Cycle(s,v)
          G = G - (s,v)
    }
    G = G - v
    H = Biconnected(G)
    for each h in H {
          FindCycles(h)
    }
}
```

Figure E.2 The main procedure for finding all cycles in a graph.

A description of the Biconnected() procedure for finding the biconnected components of an undirected graph can be found in [MaD76].

Appendix F: Extension to FCRIP Formulation

In Chapter 10, we developed the FCRIP formulation of the multi-ring network design problem based on the assumption that there is an ADM at each node of a candidate ring and there are no restrictions on the locations at which inter-ring transitions take place. In practice, the lowest cost designs may include rings with glassthroughs at some nodes. In this appendix, we extend the basic FCRIP formulation to include decision variables for the location of ADMs within each candidate ring. We also discuss how the formulation can be extended to include UPSR rings.

F.1 ADM Locations

To extend the problem formulation to include specification of the ADM locations we replace the decision variables G_{ij} in Chapter 10 with decision variables for the quantity of demand on route *i* that is carried on each span *s* of (stacked) ring *j*, G_{ijs}. We also introduce decision variables V_{jn} such that $V_{jn} = 1$ if an ADM is present at node *n* on (stacked) ring *j* and 0 otherwise. Using this notation, the IP formulation can be expressed as:

Minimize:
$$\sum_{j \in J} c_j \cdot X_j + \sum_{j \in J} \sum_{n \in N(j)} f_j \cdot V_{jn} + \sum_{i \in I} \sum_{j \in J(i)} \sum_{s \in S(i,j)} b_{ij} \cdot G_{ijs} \qquad (1)$$

Subject to:
$$\sum_{i \in I(k)} F_i = d_k, \qquad \forall k \in K \qquad (2)$$

$$\sum_{j \in J(s)} \sum_{s \in S(i,j)} G_{ijs} = F_i, \qquad \forall i \in I, \forall s \in S(i) \qquad (3)$$

$$\sum_{i \in I(s)} G_{ijs} \le m_j \cdot X_j, \qquad \forall j \in J, \forall s \in S(j) \qquad (4)$$

$$G_{ijs} - G_{ij(s+1)} \le V_{jn} \cdot F_i, \qquad \forall i \in I, \forall j \in J(i), \forall s \in S(i,j) \qquad (5)$$

$$G_{ij(s+1)} - G_{ijs} \le V_{jn} \cdot F_i, \qquad \forall i \in I, \forall j \in J(i), \forall s \in S(i,j) \qquad (6)$$

$$X_j \ge 0, \text{ integer}, \qquad \forall j \in J \qquad (7)$$

$$F_i \ge 0, \text{ integer}, \qquad \forall i \in I \qquad (8)$$

$$G_{ijs} \ge 0, \text{ integer}, \qquad \forall i \in I, \forall j \in J(i) \qquad (9)$$

where spans *s* and *(s+1)* are incident on node *n* and are given in sequence from entry to exit node. Constraints (5) and (6) ensure that no demand can be added or dropped at node *n* if $V_{jn} = 0$, i.e. $G_{ijs} = G_{ij(s+1)}$. Note that if more than one copy of a candidate ring is required, there is still the problem of deciding which rings required an ADM at the specified nodes.

F.2 UPSR Rings

To add UPSR rings to the FCRIP formulation, we denote the subset of candidate rings that are UPSRs as $J_{UPSR} \subseteq J$ and those that are BLSRs as $J_{BLSR} \subseteq J$. Constraint set (4) is then modified as follows:

$$\sum_{i \in I(s)} G_{ijs} \leq m_j \cdot X_j, \qquad \forall j \in J_{BLSR}, \forall s \in S(j) \qquad (10)$$

and an additional constraint is added to model the consumption of UPSR capacity, as follows:

$$\sum_{i \in I(j)} G_{ijs} \leq m_j \cdot X_j, \qquad \forall j \in J_{UPSR} \qquad (11)$$

Constraint (10) ensures that the total flow on each span of a (stacked) BLSR ring does not exceed its aggregate capacity and constraint (11) ensures that the total flow on each (stacked) UPSR ring does not exceed its aggregate capacity.

Appendix G: AMPL Formulations

G.1 SCIP Formulation

```
# This AMPL model finds the min-cost set of
# rings (with a given module size) that covers
# the working load assigned to each span in the
# network graph.
#
# By: G.D. Morley, May 1, 1998.
#

#--- Sets ---
set SPAN;  # set of spans in the network.
set CYCLE; # set of cycles in the network.
set CYCLE_SPAN within {CYCLE,SPAN};  # subset of spans in each cycle.
set TECH;  # ring technologies (modularities) under consideration.
set RING := {CYCLE,TECH};  # combinations of cycle and technology.

#--- Parameters ---
param moduleSize{TECH} >= 0; # the module size of each technology.
param cost{RING} >= 0;       # fixed cost of each ring.
param working{SPAN} >= 0;    # working load on each span

#--- Decision variables ---
var Number{RING} integer >=0, <=10;  # number of copies of each ring.

#--- Objective function ---
minimize total_cost:
      sum{(i,j) in RING} cost[i,j] * Number[i,j];

#--- Constraints ---
subject to span_coverage {j in SPAN}:
      sum {(i,j) in CYCLE_SPAN, k in TECH: (i,k) in RING}
            moduleSize[k] * Number[i,k] >= working[j];
```

G.2 FCRIP Formulation

```
# This AMPL model finds a set of rings (with a given
# module size) and a routing for each demand that minimizes
# the total fixed charge and routing costs of the design.
#
# By: G.D. Morley, May 1, 1998.
#

#--- Sets ---
set SPAN;   # set of spans in the network.
set CYCLE;  # set of cycles in the network.
set CYCLE_SPAN within {CYCLE,SPAN}; # subset of spans in each cycle.
set TECH;   # ring technologies (modularities) under consideration.
set RING := {CYCLE,TECH};     # combinations of cycle and technology.
set DEMAND; # set of network demands.
set ROUTE;  # a set of routes.
set ROUTE_SPAN within {ROUTE,SPAN}; # subset of spans in each route.
set DEMAND_ROUTE within {DEMAND,ROUTE};# routes assoc. with each demand.
set ROUTE_CYCLE := {i in ROUTE, j in CYCLE: card({(i,l) in ROUTE_SPAN}
inter {(j,l) in CYCLE_SPAN}) > 0}; routes that intersect each cycle.

#--- Parameters ---
param moduleSize{TECH} >= 0;  # the module size of each technology.
param cost{RING} >= 0;        # fixed cost of each ring.
param unitCost >= 0;          # cost per inter-ring transition.
param termCost >= 0;          # termination cost per demand.
param quantity{DEMAND} >= 0;  # quantity for each demand pair.

#--- Decision variables ---
var Number{RING} integer >= 0, <= 10;# number of copies of each ring.
var Flow{ROUTE} integer >= 0, <= 100;# flow over each route.
var RingFlow{ROUTE_CYCLE,TECH} integer >= 0, <= 100;# flow on each ring.

#--- Objective function ---
minimize total_cost:
      sum {(j,h) in RING} cost[j,h] * Number[j,h] +
      sum {(i,j) in ROUTE_CYCLE, h in TECH} unitCost * RingFlow[i,j,h] +
      sum {i in ROUTE} termCost * Flow[i];

#--- Constraints ---
subject to total_flow {k in DEMAND}:
      sum {(k,i) in DEMAND_ROUTE} Flow[i] = quantity[k];

subject to demand_served {(i,l) in ROUTE_SPAN}:
      sum{(j,l) in CYCLE_SPAN, h in TECH: (j,h) in RING}
            RingFlow[i,j,h] >= Flow[i];

subject to capacity {(j,l) in CYCLE_SPAN, k in TECH: (j,k) in RING}:
      sum {(i,l) in ROUTE_SPAN} RingFlow[i,j,k] <= moduleSize[k] *
            Number[j,k];
```

G.3 FDIP Formulation

```
# This AMPL model finds a min-cost set of rings (with a given
# module size) that carries each demand at least once.
#
# By: G.D. Morley, May 1, 1998.
#

#--- Sets ---
set SPAN;   # set of spans in the network graph.
set CYCLE;  # set of cycles in the network graph.
set TECH;   # ring technologies (modularities) under consideration.
set DEMAND; # set of network demands.
set CYCLE_SPAN within {CYCLE,SPAN}; # subset of spans in each cycle.
set RING within {CYCLE,TECH}; # combinations of cycle and technology.
set RING_DEMAND within {RING,DEMAND};# demands that intersect each ring.

#--- Parameters ---
param moduleSize{TECH} >= 0; # the module size of each technology.
param cost{RING} >= 0;  # the fixed cost of each ring.

#--- Decision variables ---
var Number{RING} binary;      # true if the ring is in the design.

#--- Objective function ---
minimize total_cost:
      sum {(i,j) in RING} cost[i,j] * Number[i,j];

#--- Constraints ---
subject to demand_served {k in DEMAND}:
      sum {(i,j,k) in RING_DEMAND} Number[i,j] >= 1;
```

G.4 LBIP Formulation

```
# AMPL Model for calculating the lower bound
# on the total BLSR capacity (based on several
# conditioning arguments) required to cover
# the working load assigned to each span of the
# network.
#
# By: G.D. Morley, Jan. 21, 2000
#

#--- Sets ---
set NODE; # nodes
set SPAN; # spans
set NODE_SPAN within {NODE,SPAN};   # spans incident on each node.

#--- Parameters ---
param moduleSize >= 0;        # the working capacity of the ring.
param working{SPAN} >= 0;     # the working load on each span.
param distance{SPAN} >= 0;    # the length of each span.

#--- Decision variables ---
var Modules{SPAN} integer >=0;# the number of ring modules per span.
var Rings{NODE} integer >=0;  # the number of rings at each node.

#--- Objective function ---
minimize total_modules:
      sum{j in SPAN} Modules[j];

#--- Constraints ---
subject to span_coverage {j in SPAN}:
      moduleSize * Modules[j] >= working[j];

subject to ring_balance {i in NODE, (i,j) in NODE_SPAN}:
      sum {(i,k) in NODE_SPAN} Modules[k] >= 2 * Modules[j];

subject to ring_parity {i in NODE}:
      sum {(i,j) in NODE_SPAN} Modules[j] = 2 * Rings[i];
```

G.5 LBRIP Formulation

```
# AMPL Model for calculating the lower bound
# on the total BLSR capacity (based on several
# conditioning arguments) required to cover
# the working assigned to each span of the
# network. This formulation also optimizes
# the routing of demands.
#
# By: G.D. Morley, Aug. 18, 2000
#

#--- Sets ---
set NODE; # set of nodes in the network.
set SPAN; # set of spans in the network.
set NODE_SPAN within {NODE,SPAN};   # spans incident on each node.
set DEMAND; # set of network demands.
set ROUTE;  # set of routes.
set ROUTE_SPAN within {ROUTE,SPAN}; # spans in each route.
set DEMAND_ROUTE within {DEMAND,ROUTE};# routes assoc. with each demand.
set TECH;    # ring technologies (modularities) under consideration.

#--- Parameters ---
param moduleSize{TECH} >= 0;  # the working capacity of each technology.
param quantity{DEMAND} >= 0;  # the quantity
param distance{SPAN} >= 0;    # the length of each span.

#--- Decision variables ---
var Modules{SPAN,TECH} integer >=0; # num. of ring modules on each span.
var Rings{NODE,TECH} integer >=0;   # number of rings at each node.
var Flow{ROUTE} integer >=0;        # the flow over each route.

#--- Objective function ---
minimize cost:
      sum{j in SPAN, k in TECH} distance[j] * Modules[j,k];

#--- Constraints ---
subject to span_coverage {j in SPAN}:
      sum {k in TECH} moduleSize[k] * Modules[j,k] >=
            sum {(i,j) in ROUTE_SPAN} Flow[i];

subject to total_flow {k in DEMAND}:
      sum {(k,i) in DEMAND_ROUTE} Flow[i] = quantity[k];

subject to ring_balance {i in NODE, (i,j) in NODE_SPAN, k in TECH}:
      sum {(i,l) in NODE_SPAN} Modules[l,k] >= 2 * Modules[j,k];

subject to ring_parity {i in NODE, k in TECH}:
      sum {(i,j) in NODE_SPAN} Modules[j,k] = 2 * Rings[i,k];
```